IMPLEMENTASI *MOBILE AD-HOC NETWORK* (MANET) DENGAN PROTOKOL AODV PADA PERANGKAT BERBASIS NRF24L01

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Arief Indra Rivaldy Permana
NIM: 135150301111012

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI *MOBILE AD-HOC NETWORK* (MANET) DENGAN PROTOKOL AODV PADA PERANGKAT BERBASIS NRF24L01

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Arief Indra Rivaldy Permana
NIM: 135150301111012

Skripsi ini telah diuji dan dinyatakan lulus pada
1 Agustus 2018

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Sabriansyah Rizqika Akbar, S.T, M.Eng	Rakhmadhany Primananda, S.T, M.Kom
NIP: 1982080920121004	NIK: 2016098604061001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 10 Agustus 2018

Arief Indra Rivaldy Permana

NIM: 135150301111012

KATA PENGANTAR

Alhamdulillahhirabbil'alamin, puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya kepada penulis, sehingga dapat menyelesaikan skripsi berjudul "Implementasi *Mobile Ad-Hoc Network* (Manet) dengan Protokol AODV pada Perangkat Berbasis NRF24L01" ini. Sholawat serta salam tak lupa saya curahkan kepada Nabi Muhammad SAW sebagai pembawa *Rahmatan Lil'Alaamin.*

Skripsi ini merupakan salah satu mata kuliah yang wajib ditempuh sebagai syarat kelulusan dan memperoleh gelar Sarjana di Fakultas Ilmu Komputer Universitas Brawijaya. Dalam proses penyusunan laporan ini penulis mendapatkan banyak bantuan dari berbagai pihak baik secara moril maupun materiil. Dalam kesempatan ini penulis ingin mengucapkan terimakasih yang sebesar-besarnya kepada:

1. Bapak Sabriansyah Rizqika Akbar, ST., M.Eng. selaku dosen Pembimbing Pertama skripsi ini.

2. Bapak Rakhmadhany Primananda, S.T, M.Kom selaku dosen Pembimbing Kedua skripsi ini.

3. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer, Universitas Brawijaya, Malang.

4. Bapak Heru Nurwasito, Ir., M.Kom. selaku Dekan I Bidang Akademik Fakultas Ilmu Komputer, Universitas Brawijaya, Malang.

5. Bapak Tri Astoto Kurniawan, S.T.,M.T., Ph.D selaku Ketua Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya, Malang.

6. Kedua orang tua penulis, Muhammad Nuceh Hassan dan Umi Salamah serta keluarga, yang selalu senantiasa mendoakan kelancaran skripsi ini serta sebagai penyemangat penulis.

7. (Alm.) Nanda Cahyo Wirawan S.Kom atas dukungan, semangat, waktu, dan kenangan yang telah penulis rasakan ketika dalam proses penyelesaian laporan skripsi ini.

8. Sahabat-sahabatku Budi, Nazzun, Nabilah, Yayak, teman-teman "Club Kerang Ajaib", "UB48 Community & JKT48 Fans UB", "FIBeats", "Dai-Toku UB", teman-teman sedari Semester 1 "SisKom B 2013 & MAOU Brawijaya", teman-teman sedari Skripsian "Ruang Lab Riset Filkom C1.7", Geraldo dan Eko sebagai teman sesama skripsian yang saling membantu, Ibad, Andyan, Chuni atas dukungan yang sedari awal hingga akhir diberikan, penulis haturkan terimakasih atas doa, semangat, kerjasama, canda tawa dan dukungannya.

9. Rekan-rekan Program Studi Teknik Komputer 2013 dan orang-orang yang selalu mendukung dan mendoakan penulis yang tidak dapat penulis sebutkan satu per satu, yang selalu memberikan semangat dan dukungan sehingga skripsi ini

selesai. Sekali lagi penulis haturkan terima kasih atas semua doa dan dukungan baik dalam bentuk materiil maupun non materiil.

Penulis menyadari bahwa dalam penyusunan laporan Skripsi ini masih banyak kekurangan baik dari segi susunan maupun isi. Oleh sebab itu, penulis dengan senang hati dan terbuka menerima kritik dan saran yang membangun sebagai pedoman perbaikan kedepannya dan berharap adanya pengembangan lebih lanjut oleh pihak-pihak terkait.

Semoga laporan ini dapat memberikan manfaat dan referensi untuk melakukan penelitian selanjutnya sebagai langkah penyempurnaan sistem. Jangan sungkan untuk menghubungi penulis apabila ingin berdiskusi mengenai pengembangan lebih lanjut mengenai skripsi ini.

Malang, 10 Agustus 2018

Penulis

masaldy07@yahoo.co.id

ABSTRAK

Implementasi protokol AODV pada jaringan *Mobile Ad-Hoc Network* (Manet) yang sudah ada pada umumnya dilakukan menggunakan perangkat lunak simulasi. Penggunaan perangkat lunak simulasi yang sering digunakan adalah NS-3, OPNET atau OMNET. Kekurangan dari penerapan pada perangkat lunak simulasi tersebut adalah belum ditemukan adanya pengujian dengan skenario kondisi menggunakan hambatan, dikarenakan selalu dalam kondisi ideal. Selain itu, minimnya penerapan protokol AODV pada jaringan manet menggunakan *real device*. Untuk mengatasi kekurangan yang dihadapi oleh perangkat lunak simulasi tersebut, pada penelitian ini diterapkan implementasi *mobile ad-hoc network* (manet) dengan menggunakan protokol AODV pada perangkat keras nRF24L01 untuk melakukan pengiriman data yang diinput oleh *user* serta melakukan uji fungsionalitas sistem dengan menjalankannya disertai dengan adanya hambatan berupa pintu, dinding, dan kaca. Dari hasil penelitian, didapatkan hasil bahwa setiap *node* dapat saling berkomunikasi pada jaringan *mobile ad-hoc network* dengan menggunakan Arduino UNO sebagai mikrokontroler dan modul *wireless* nRF24L01. Protokol AODV sebagai metode *routing* protokol antar *node* berhasil diimplementasikan. Setiap *node* juga telah berhasil mengirim pesan dari proses pengujian yang dilakukan. Sedangkan dari 15 kali pengujian fungsionalitas sistem, dimana skenarionya adalah *node* statis tanpa hambatan, *node* bergerak dinamis tanpa adanya hambatan dan *node* bergerak dinamis dengan adanya hambatan berupa pintu, dinding, atau kaca, semua *node* melakukan pengiriman dan penerimaan data dengan persentase keberhasilan sebesar 86,6%. Hal ini membuktikan bahwa pengujian fungsionalitas sistem telah berhasil diterapkan dengan baik.

Kata kunci : *Mobile Ad-Hoc Network,* AODV*,* NRF24L01*,* Protokol *Routing,* Arduino

ABSTRACT

Implementation of AODV *protocol on Mobile Ad-Hoc Network (*Manet*) is basically done using simulation software. Using software like* NS-3*,* OPNET *or* OMNET*. Disadvantage of those simulation software is there is no obstacles because of the ideal condition. The lack of* AODV *protocol implementation on the manet is using real devices. To overcome the mistakes made by the software, in this study applied the implementation of mobile ad-hoc network (*manet*) by using* AODV *protocol on* nRF24L01 *hardware to send data inputted by user and to test the system functionality by running it with any obstacles like doors, walls, and glasses. From the results of this research, is that each node can communicate with each other on ad-hoc mobile network using* Arduino UNO *as microcontroller and* nRF24L01 *wireless module. The* AODV *protocol as an inter-node protocol routing method has been successfully implemented. Each node has also successfully process performed. On* 15 *times the system test functionality, where the scenario is a static node without any obstacles, the node moves dynamically without any obstacles and the node moves dynamically with the obstacles of doors, walls, or glasses, all nodes was sending and receiving data with the percentage of success is* 86.6%*. This proved that the system functionality has been successfully implemented properly.*

Keyword: Mobile Ad-Hoc Network, AODV*,* NRF24L01*, Routing Protocol,* Arduino

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Mobile Ad-Hoc Network (MANET) merupakan jaringan *wireless multihop* yang terdiri dari kumpulan *mobile node* yang bersifat dinamik. Sistem yang digunakan pada MANET bersifat yaitu mampu mengatur diri sendiri serta dibentuk oleh sekumpulan *node* atau terminal yang dihubungkan oleh jalur-jalur nirkabel. Dalam suatu jaringan, konektivitas beberapa *node* dapat menghilang karena jarak yang terlalu jauh dan muncul *node* baru dalam satu waktu dikarenakan pergerakan bebas *node-node* tersebut (Benardi, 2009).

Jika *node-node* bergerak bebas maka akan menyebabkan topologi jaringan berubah. Keadaan ini mengakibatkan jaringan tidak memiliki infrastruktur atau dikenal dengan istilah jaringan *ad-hoc*. Pesan yang dikirim dalam lingkungan jaringan ini akan terjadi antara dua *node* dalam cakupan transmisi masing-masing. Dan juga secara tidak langsung dihubungkan oleh *multiple hop* melalui beberapa *node* perantara jika kedua *node* itu tidak dapat berhubungan atau berada di luar jangkauannya. *Node* pada MANET tidak hanya berperan sebagai pengirim atau penerima data saja, namun dapat juga difungsikan sebagai penghubung *node* yang lain. Untuk mengatur seluruh proses *routing* pada topologi MANET tidak memerlukan *router/node*, karena setiap *device* berfungsi sebagai *router* untuk menentukan arah yang akan di tentukan (Amilia, et al., 2014). Sehingga pada proses komunikasi pada jaringan MANET dibutuhkan sebuah aturan berupa protokol *routing* untuk menentukan rute pengiriman paket, agar *node* dapat mengirimkan paket data yang dibutuhkan oleh jaringan MANET tersebut.

AODV adalah *routing protocol* yang dirancang untuk jaringan *ad-hoc mobile*. *Route request* (RREQ), *route reply* (RREP) dan *route error* (RERR) merupakan jenis-jenis pesan yang ditentukan oleh AODV. Pesan-pesan tersebut dikirim menggunakan pengalamatan IP. Dalam pengalamatan IP, pesan tersebut ditambahkan *header* yang berfungsi untuk menentukan alamat yang akan dituju. Setelah sampai di penerima, IP *header* tersebut akan dipecah untuk mengetahui isi pesan yang dikirim. Pesan yang disebar memiliki waktu hidup (*time to live*) yang dibawa oleh *header* pada IP (Perkins, et al., 2003). Selama koneksi rute dari pengirim ke penerima telah valid, AODV tidak melakukan pencarian rute lagi. Sebaliknya ketika diperlukan rute ke penerima yang baru, maka pengirim akan menyebarkan pesan *route request* (RREQ) secara *broadcast* ke semua *node* tetangga. *Node* tetangga yang menerima RREQ akan mengirim pesan balasan berupa RREP jika *node* tersebut adalah penerima atau memiliki rute ke penerima. *Node* yang mengetahui rute ke penerima disebut *node* penghubung. Baik *node* penghubung dan penerima akan menyimpan informasi baru yang dibawa oleh RREQ, kemudian mengirim RREP ke pengirim. Setiap *node* yang dilewati RREP akan membentuk suatu rute sendiri menuju pengirim. Jadi melalui pesan RREP inilah rute *end to end* antara pengirim ke penerima terbentuk. Pengirim akan menerima pesan RREP yang berisi informasi tentang alamat pengirim, alamat penerima, nomor urutan dari penerima, *hop count* dan waktu hidup pesan. Sumber akan

mengganti rute apabila rute yang baru memiliki nomor urutan yang lebih besar dan *hop count* yang lebih sedikit dari rute yang ada saat ini. Selama rute terbentuk, setiap *node* dalam jaringan memantau kondisi *link* di depannya untuk mengantisipasi adanya kerusakan. Apabila sebuah rute mengalami kerusakan atau terputus, maka *node* yang terhubung pada link tersebut akan memberitahukan ke seluruh *node* bahwa rute tersebut rusak. Kemudian *node* yang bersangkutan akan menyebarkan RERR ke seluruh *node* tetangga hingga ke pengirim. RERR mengindikasikan bahwa penerima tidak dapat dicapai melalui rute yang rusak. Oleh karena itu pengirim harus menyebarkan RREQ secara ulang (Perkins, et al., 2003).

Permasalahan yang akan diangkat dalam skripsi ini, adalah dikarenakan belum adanya penerapan menggunakan *real device* atau seperti dalam penelitian ini menggunakan nRF24L01. Selain itu dari penelitian sebelumnya yang penulis baca dan jadikan acuan, penerapan pada simulator belum ada pengujian dengan kondisi menggunakan hambatan, selalu dalam kondisi ideal.

Maka berdasarkan permasalahan yang muncul pada referensi latar belakang tersebut, penulis akan melakukan penelitian dengan judul "Implementasi Mobile *Ad-Hoc Network* (Manet) Dengan Protokol AODV pada Perangkat Berbasis nRF24L01", yang disertai dengan pengujian implementasi sistem dengan pengiriman data input dari *user* dengan pengujian fungsionalitasnya apabila dijalankan dengan hambatan berupa pintu, dinding, dan kaca.

1.2 Rumusan masalah

Penelitian ini dirumuskan dalam beberapa hal berikut ini:

1. Bagaimana rancangan dan implementasi *mobile ad-hoc network* (manet) agar dapat dijalankan di perangkat berbasis NRF24L01?
2. Bagaimana rancangan dan implementasi protokol AODV pada perangkat berbasis NRF24L01?
3. Bagaimana hasil pengujian implementasi manet dengan protokol AODV pada perangkat berbasis NRF24L01?
4. Bagaimana hasil pengujian fungsionalitas sistem manet dengan protokol AODV pada perangkat berbasis NRF24L01?

1.3 Tujuan

Penelitian ini bertujuan:

1. Untuk merancang dan mengimplementasikan *mobile ad-hoc network* (manet) agar dapat dijalankan di perangkat berbasis NRF24L01
2. Untuk merancang dan mengimplementasikan protokol AODV pada perangkat berbasis NRF24L01
3. Untuk menguji implementasi manet dengan protokol AODV pada perangkat berbasis NRF24L01
4. Untuk menguji fungsionalitas sistem manet dengan protokol AODV pada perangkat berbasis NRF24L01

1.4 Manfaat

Penelitian ini diharapkan mampu memberi manfaat dan berkontribusi langsung dalam membantu memberikan efisiensi dan manfaat yang sangat besar, terutama dalam hal investasi perangkat yang membentuk jaringan semacam ini. Terbentuknya model jaringan MANET akan memungkinkan perancangan dan evaluasi terhadap jaringan bisa dilakukan tanpa harus secara fisik membangun infrastrukturnya yang besar dan kompleks terlebih dahulu. Selain itu, penelitian ini diharapkan mampu memberi manfaat dan berkontribusi langsung dalam membantu mendapatkan peningkatan pemerataan koneksi jaringan *wireless* dan variasi komunikasi di jaringan yang sudah ada dan diterapkan hingga saat ini.

1.5 Batasan masalah

Permasalahan dalam penelitian dibatasi dalam beberapa hal berikut ini:
1. Penelitian ini dilakukan menggunakan perangkat keras Mikrokontroler Arduino UNO
2. Penelitian ini terfokus pada implementasi topologi Manet dengan protokol AODV
3. Paket data yang dikirim maksimal 32 *char*.
4. Terdapat satu *node* yang berfungsi sebagai *node base* sekaligus menjadi penampung data dari 3 *node client* yang terhubung secara *wireless.*
5. Pada *node client* menggunakan mikrokontroler Arduino UNO dan modul *wireless* nRF24L01.
6. Pada *node base* menggunakan mikrokontroler Arduino UNO dan modul *wireless* nRF24L01.
7. Pengujian dilakukan di dalam dan diluar ruangan dengan kondisi ada dan tidak ada penghalang diantara *node*.
8. Peletakan setiap *node* tidak lebih dari jarak 10 meter.
9. Hambatan yang digunakan adalah pintu, dinding, dan kaca.
10. Data yang dikirim oleh *node client* berupa nilai input yang didapatkan melalui ketikan oleh user pada serial monitor di *node base.*

1.6 Sistematika pembahasan

Penyusunan skripsi ini menggunakan kerangka pembahasan yang tersusun sebagai berikut:

BAB I PENDAHULUAN
Membahas tentang latar belakang masalah dari penelitian, rumusan masalah, batasan – batasan masalah, metode penelitian, tujuan serta sistematika penulisan dari penelitian ini.
BAB II TINJAUAN PUSTAKA
Berisi tinjauan pustaka yang berisi berbagai referensi dan landasan teori yang menjadi dasar dari penelitian ini. Pada bab ini akan diterangkan secara detail sesuai informasi serta studi pustaka yang diperoleh peneliti berkaitan dengan topik penelitian. Bab ini juga menjadi acuan peneliti untuk melakukan tahapan – tahapan penelitian.

BAB III METODOLOGI PENELITIAN

Berisi mengenai metode yang digunakan dalam skripsi ini, serta kebutuhan akan hardware maupun software untuk mendukung penelitian.

BAB IV REKAYASA PRASYARAT / KEBUTUHAN

Bab ini menjelaskan proses analisis kebutuhan pengguna dan kebutuhan sistem serta pemilihan metode.

BAB V PERANCANGAN DAN IMPLEMENTASI

Bab ini menjelaskna proses perancangan berdasarkan kebutuhan yang telah didapatkan dan implementasi atau tahap-tahap pembuatan model jaringan Manet berdasarkan kebutuhan dan perancangan yang telah didefinisikan.

BAB VI PENGUJIAN

Bab ini berisi pengujian model system yang telah dibangun.

BAB VII KESIMPULAN DAN SARAN

Berisi kesimpulan dari hasil penelitian serta saran – saran guna penelitian lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Pada penelitian ini, pustaka sebagai referensi berasal dari jurnal yang terkait dengan topik manet dan protokol AODV, yaitu menggunakan referensi penelitian yang berjudul "Simulasi Model Jaringan *Mobile Ad-Hoc* (Manet) dengan NS-3" dari Dedy Irawan. Dilanjutkan dengan penelitian "Perbandingan Kinerja Protokol AODV Dengan OLSR Pada Manet" yang dilakukan oleh Wahyu Edy Seputra. Ditambah dengan dasar teori terkait berdasarkan latar belakang dan rumusan masalah, yaitu *Mobile Ad-Hoc Network* (Manet), Protokol AODV, pemrograman Arduino, dan modul komunikasi *wireless* nRF24L01.

2.1 Tinjauan Pustaka

Tinjauan pustaka membahas tentang beberapa penelitian yang sudah ada dan berkaitan dengan penelitian yang diusulkan. Tinjauan pustaka ini sebagai dasar penulis dalam mengerjakan penelitian ini.

Tabel 2.1 Tabel Perbandingan Paper dengan Penelitian yang Dilakukan

No.	Judul *Paper*	Nama	Tahun	Persamaan	Perbedaan	
					Penelitian Terdahulu	Rencana Penelitian
1.	Simulasi Model Jaringan *Mobile Ad-Hoc* (Manet) dengan NS-3	Dedy Irawan	2016	Implementasi Jaringan *Mobile Ad-Hoc Network (Manet)*	Implementasi Jaringan *Manet* menggunakan *Simulator NS-3*	Implementasi Jaringan *Manet* menggunakan perangkat keras *nRF24L01*
2.	Perbandingan Kinerja Protokol AODV Dengan OLSR Pada Manet	Wahyu Edy Seputra	2011	Implementasi Protokol AODV pada Jaringan *Manet*	Implementasi Protokol AODV pada Jaringan *Manet* menggunakan *Simulator OPNET* dengan jumlah *node* 50 dan 100	Implementasi Protokol AODV pada Jaringan *Manet* menggunakan perangkat keras *nRF24L01* dengan jumlah *node* 4

2.2 Dasar Teori

Dalam sub bab ini akan dijelaskan referensi dasar teori sebagai pengetahuan tentang komponen dan teknologi yang digunakan meliputi *Mobile Ad-Hoc Network* (Manet), Protokol AODV, pemrograman Arduino, dan modul komunikasi *wireless* nRF24L01.

2.2.1 MANET

2.2.1.1 Definisi MANET

Mobile *Ad-hoc Network* (MANET) yaitu sebuah jaringan *wireless* dari *mobile-mobile node* yang tidak memiliki *router* tetap. Kumpulan dari beberapa *wireless node* ini dapat di *set-up* secara dinamis dimana saja dan kapan saja tanpa menggunakan infrastruktur jaringan yang ada. Bisa dibilang juga juga merupakan jaringan sementara yang dibentuk oleh beberapa *mobile node* tanpa adanya pusat administrasi dan infrastruktur kabel. Pada MANET, *mobile host* yang terhubung dengan *wireless* dapat bergerak bebas dan juga berperan sebagai *router*, yang bertanggung jawab untuk mencari dan menangani rute ke setiap *node* di dalam jaringan. MANET yang ingin berinterkoneksi dengan *fixed host* harus melewati *gateway* terlebih dahulu.

Apabila *mobile node* ingin berinterkoneksi dengan *fixed host* maka rute ke *gateway* harus segera ditemukan. Metode *gateway discovery* yang ada, menawarkan kemudahan *mobile node* untuk bisa menemukan dan melakukan hubungan dengan *fixed node*.

Adakalanya pencarian *gateway* diawali oleh *mobile node* sendiri yang diistilahkan dengan metode *reactive gateway discovery*, atau bisa juga diawali oleh *gateway* sebagai pintu gerbang interkoneksi yang diistilahkan dengan *proactive gateway discovery*. Adaptasi terhadap kondisi jaringan yang tidak selalu bisa ditangani oleh metode reaktif ataupun proaktif menghasilkan metode gabungan / *hybrid*.

Terdapat beberapa perbedaan antara jaringan *ad-hoc* dengan jaringan yang memiliki infrastruktur, antara lain:

- *Peer-to-Peer*, yaitu komunikasi antara dua *node* dalam satu *hop*.
- *Remote-to-Remote*, yaitu komunikasi antar dua *node* diluar satu *hop*, namun masih tetap mengelola kestabilan rute di antara keduanya.
- *Dynamic Traffic*, terjadi ketika *node* bergerak, maka rute harus dikonstruksi ulang. Ini merupakan hasil dari tingkat konektifitas yang rendah.

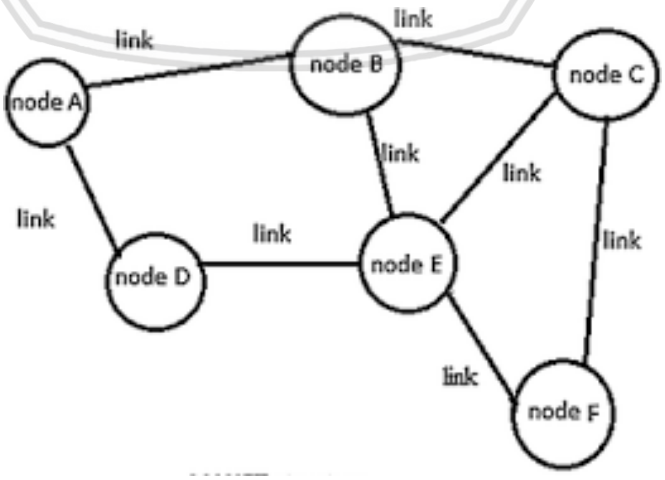
2.2.1.2 Karakteristik MANET

MANET terdiri dari *mobile platform* (seperti *router* dan perangkat *wireless*) dalam hal ini disebut dengan "*node*" yang bebas berpindah-pindah ke mana saja. *Node* tersebut bisa saja berada di pesawat, kapal, mobil dan dimana saja.

Setiap *node* dilengkapi dengan *transmitter* dan *receiver wireless* menggunakan antena atau sejenisnya yang bersifat *omnidirectional* (*broadcast*), *highly directional* (*point to point*), memungkinkan untuk diarahkan, atau kombinasi dari beberapa hal tersebut. *Omnidirectional* maksudnya adalah gelombang radio dipancarkan ke segala arah oleh perangkat *transmitter wireless*. Sedangkan *highly directional* adalah gelombang dipancarkan ke satu arah tertentu.

Selain karakteristik di atas, *Mobile Ad*-h*oc Network* (MANET) juga memiliki beberapa karakteristik yang lebih menonjol, antara lain:

- Topologi yang dinamis : *Node* pada MANET memiliki sifat yang dinamis, yaitu dapat berpindah-pindah kemana saja. Maka topologi jaringan yang bentuknya adalah loncatan antara *hop* ke *hop* dapat berubah secara tidak terpola dan terjadi secara terus menerus tanpa ada ketetapan waktu untuk berpindah. Bisa saja didalam topologi tersebut terdiri dari *node* yang terhubung ke banyak *hop* lainnya, sehingga sangat berpengaruh secara signifikan terhadap susunan topologi jaringan.
- Otonomi : Setiap *node* pada MANET berperan sebagai *end-user* sekaligus sebagai *router* yang menghitung sendiri *route-path* yang selanjutnya akan dipilih.
- Keterbatasan *bandwidth* : Link pada jaringan *wireless* cenderung memiliki kapasitas yang rendah jika dibandingkan dengan jaringan berkabel. Jadi, kapasitas yang keluar untuk komunikasi *wireless* juga cenderung lebih kecil dari kapasitas maksimum transmisi. Efek yang terjadi pada jaringan yang berkapasitas rendah adalah *congestion* (kemacetan).
- Keterbatasan energi : Semua *node* pada MANET bersifat *mobile*, sehingga sangat dipastikan *node* tersebut menggunakan tenaga baterai untuk beroperasi. Sehingga perlu perancangan untuk optimalisasi energi.



Gambar 2.1 Struktur Jaringan Manet

(Sumber: https://doi.org/10.1007/978-81-322-2638-3_105)

2.2.1.3 Protokol

Terdapat berbagai jenis protokol *routing* untuk MANET yang secara keseluruhan dapat dibagi menjadi beberapa kelompok, antara lain:

a. *Proactive Routing*

Algoritma ini akan mengelola daftar tujuan dan rute terbaru masing-masing dengan cara mendistribusikan *routing table* ke seluruh jaringan, sehingga jalur lalu lintas (*traffic*) akan sering dilalui oleh *routing table* tersebut. Hal ini akan memperlambat aliran data jika terjadi restrukturisasi *routing table*. Beberapa contoh algoritma *proactive routing* adalah:

- Babel
- B.A.T.M.A.N – *Better Approach to Mobile Ad-hoc Network*
- DSDV – *Highly Dynamic Destination Sequenced Distance Vector routing protocol*
- HSR – *Hierarchial State Routing Protocol*
- IARP – *Intrazone Routing Protocol*
- LCA – *Linked Cluster Architecture*
- WAR – *Witness Aided Routing*
- OLSR – *Optimized Link State Routing Protocol*

b. *Reactive Routing*

Tipe ini akan mencari rute (*on demand*) dengan cara membanjiri jaringan dengan paket *router request*. Sehingga dapat menyebabkan jaringan akan penuh (*clogging*). Beberapa contoh algoritma *reactive routing* adalah:

- SENCAST
- *Reliable Ad-hoc On Demand Distance Vector Routing Protocol*
- *Ant-Based Routing Algorithm for Mobile Ad-hoc Network*
- *Admission Control Enabled On Demand Routing* (ACOR)
- Ariadne
- *Associativity Based Routing*
- *Ad-hoc On Demand Distance Vector* (AODV)
- *Ad-hoc On Demand Multipath Distance Vector*
- *Backup Source Routing*
- *Dynamic Source Routing* (DSR)
- *Flow State in the Dynamic Source Routing*
- *Dynamic MANET On Demand Routing* (DYMO)

c. *Flow Oriented Routing*

Tipe protokol ini mencari rute dengan mengikuti aliran yang disediakan. Salah satu pilihan adalah dengan *unicast* secara terus-menerus ketika meneruskan data saat mempromosikan *link* baru. Beberapa kekurangan tipe protokol ini adalah membutuhkan waktu yang lama untuk mencari rute yang baru. Beberapa protokol yang memiliki tipe ini adalah:

- *Interzone Routing Protocol* (IERP)
- *Lightweight Underlay Network Ad-hoc Routing* (LUNAR)
- *Signal Stability Routing* (SSR)

d. *Hybrid Routing*

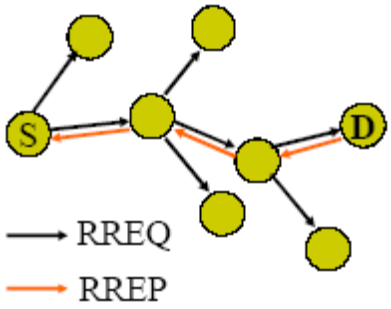
Tipe protokol ini menggabungkan antara *proactive routing* dengan *reactive routing*. Protokol untuk tipe ini adalah:

- *Hybrid Routing Protocol for Large Scale* MANET (HRPLS)
- *Hybrid Wireless Mesh Protocol* (HWMP)
- *Zone Routing Protocol* (ZRP)

2.2.2 Protokol AODV

AODV adalah *routing protocol* yang dirancang untuk jaringan *ad-hoc mobile*. *Route request* (RREQ), *route reply* (RREP) dan *route error* (RERR) merupakan jenis-jenis pesan yang ditentukan oleh AODV. Pesan-pesan tersebut dikirim menggunakan pengalamatan IP. Dalam pengalamatan IP, pesan tersebut ditambahkan *header* yang berfungsi untuk menentukan alamat yang akan dituju. Setelah sampai di penerima, IP *header* tersebut akan dipecah untuk mengetahui isi pesan yang dikirim. Pesan yang disebar memiliki waktu hidup (*time to live*) yang dibawa oleh *header* pada IP (Perkins, et al., 2003).

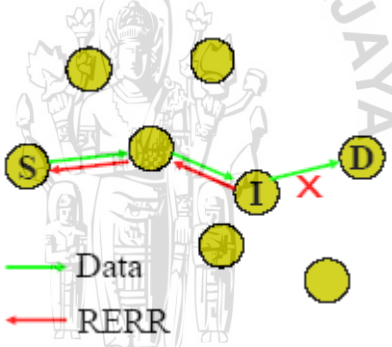
Selama koneksi rute dari pengirim ke penerima telah valid, AODV tidak melakukan pencarian rute lagi. Sebaliknya ketika diperlukan rute ke penerima yang baru, maka pengirim akan menyebarkan pesan *route request* (RREQ) secara *broadcast* ke semua *node* tetangga. *Node* tetangga yang menerima RREQ akan mengirim pesan balasan berupa RREP jika *node* tersebut adalah penerima atau memiliki rute ke penerima. *Node* yang mengetahui rute ke penerima disebut *node* penghubung. Baik *node* penghubung dan penerima akan menyimpan informasi baru yang dibawa oleh RREQ, kemudian mengirim RREP ke pengirim. Setiap *node* yang dilewati RREP akan membentuk suatu rute sendiri menuju pengirim. Jadi melalui pesan RREP inilah rute *end to end* antara pengirim ke penerima terbentuk. Pengirim akan menerima pesan RREP yang berisi informasi tentang alamat pengirim, alamat penerima, nomor urutan dari penerima, *hop count* dan waktu hidup pesan.



Gambar 2.2 Mekanisme Pencarian Rute

(Sumber: (Kopp, 2002)

Selama rute terbentuk, setiap *node* dalam jaringan memantau kondisi *link* di depannya untuk mengantisipasi adanya kerusakan. Apabila sebuah rute mengalami kerusakan atau terputus, maka *node* yang terhubung pada link tersebut akan memberitahukan ke seluruh *node* bahwa rute tersebut rusak. Kemudian *node* yang bersangkutan akan menyebarkan RERR ke seluruh *node* tetangga hingga ke pengirim. RERR mengindikasikan bahwa penerima tidak dapat dicapai melalui rute yang rusak. Oleh karena itu pengirim harus menyebarkan RREQ secara ulang (Perkins, et al., 2003).



Gambar 2.3 Mekanisme Data (*Route Update*) dan *Route Error*

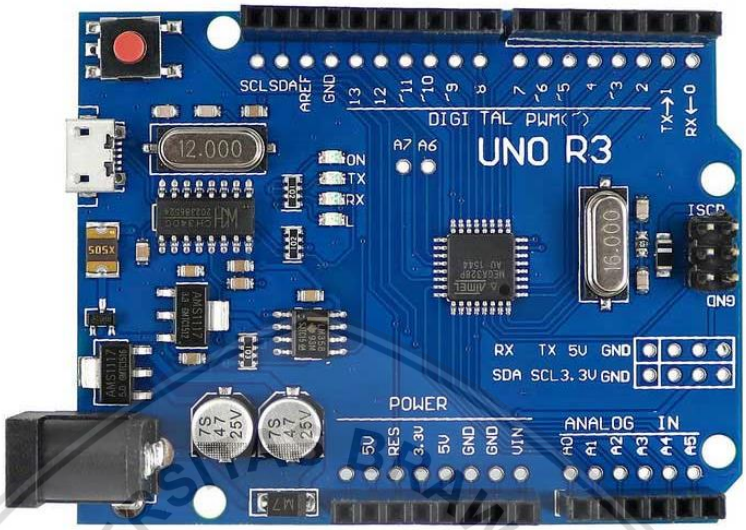
(Sumber: (Kopp, 2002)

2.2.3 Mikrokontroler

Mikrokontroler itu sendiri adalah *chip* atau IC (*integrated circuit)* yang bisa diprogram menggunakan komputer. Tujuan menanamkan program pada mikrokontroler adalah agar rangkaian elektronik dapat membaca input, memproses *input* tersebut dan kemudian menghasilkan *output* sesuai yang diinginkan. Jadi mikrokontroler bertugas sebagai 'otak' yang mengendalikan *input*, proses dan *output* sebuah rangkaian elektronik.

Arduino adalah kit elektronik atau papan rangkaian elektronik *open source* yang di dalamnya terdapat komponen utama yaitu sebuah *chip* mikrokontroler dengan jenis AVR dari perusahaan Atmel. Karena komponen utama Arduino adalah mikrokontroler, maka Arduino pun dapat diprogram menggunakan komputer sesuai kebutuhan kita. Arduino memiliki keunggulan yaitu bersifat *open-source* dan penggunanya ditujukan untuk komponen dalam *system*

embedded. Arduino UNO merupakan seri dari Arduino *family* yang berbasis ATmega328P. Untuk menuliskan program pada Arduino UNO menggunakan bahasa pemrograman C. Terdapat banyak *library* yang dapat digunakan sehingga memudahkan user dalam penggunaannya (Arduino CC, 2017).



Gambar 2.4 Board Arduino UNO

(Sumber : Arduino.cc)

Arduino UNO memililki 14 buah pin digital input/output (6 pin dapat digunakan sebagai output PWM), 6 buah pin analog input, clock speed sebesar 16MHz, jack power, tombol reset dan konektivitas USB. (Arduino CC, 2017).

a. Input/Output Pin

Seperti yang telah disebutkan sebelumnya, bahwa Arduino UNO memiliki 14 pin digital input/output. Ke-14 pin tersebut digunakan sebagai *input/output* dengan menggunakan fungsi pin *Mode(), digitalWrite(),* dan *digital(Read)*. Pin-pin tersebut bekerja pada tegangan 5V yang setiap pin-nya mendapatkan arus sebesar 20mA. Beberapa pin pada Arduino UNO memiliki fungsi khusus, diantaranya :

- ***Serial*** yang terdiri dari 2 pin, yaitu pin 0 (*RX)* dan pin 1 *(TX)* yang digunakan untuk menerima *(RX)* dan mengirim *(TX)* data serial.

- ***External Interrups*** yaitu pin 2 dan pin 3. Digunakan untuk mengaktifkan *interrupts*.

- ***PWM*** yaitu pin 3,5,6,9,10 dan 11 yang menyediakan *output* PWM-8 *bit* dengan fungsi *analogWrite()*.

- ***SPI*** yaitu pin 10 *(SS),* 11 *(MOSI),* 12 *(MISO)* dan 13 *(SCK)* yang mendukung komunikasi *SPI* dengan menggunakan *library SPI*.

- ***LED*** yaitu pin 13 yang terhubung *build-in* LED yang dikendalikan oleh *digital* pin 13.

b. Power Supply

Arduino UNO juga memiliki beberapa pin *analog*, yaitu A0 hingga A5. Board Arduino UNO diberi tegangan atau *power* yang diperoleh dari koneksi kabel *mini-USB* atau bisa juga dengan *power* eksternal. Beberapa pin *power* pada Arduino UNO antara lain :

- ***GND*** adalah pin *ground* negatif.

- ***VIN*** adalah pin yang digunakan untuk memberikan *power* langsung ke *board* Arduino Nano dengan rentang tegangan 7V – 12V.

- **Pin 5V** adalah pin *output* yang aktif jika diberikan tegangan 5V melalui *regulator*.

- **Pin 3V3** adalah pin *output* yang menyediakan tegangan sebesar 3.3V melalui *regulator*.

- **IOREF** adalah pin yang menyediakan referensi tegangan mikrokontroler. Pin ini digunakan pada *board shield* untuk mendapatkan tegangan yang sesuai antara 5V atau 3V.

c. Memori

Arduino UNO pada prinsipnya menggunakan *chip ATmega328P* yang memiliki memori sebesar 32KB dimana 0.5KB dari total memori digunakan untuk *bootloader*. Jumlah SRAM 2KB dan EEPROM 1KB. EEPROM dapat dibaca-tulis dengan menggunakan EEPROM *library* saat melakukan pemrograman. (Arduino CC, 2017)

d. Komunikasi

Arduino UNO menyediakan beberapa macam alternatif untuk dapat berkomunikasi dengan komputer atau dengan mikrokontroler lainnya. Arduino UNO menyediakan komunikasi *serial* UART TTL (5V) yang tersedia pada pin 0 *(RX)* dan pin 1 *(TX)*. Pada bagian komunikasi, terdapat Perangkat lunak Arduino IDE untuk dapat menampilkan data *serial* dengan menggunakan fasilitas *Serial Monitor* (Arduino CC, 2017).

2.2.4 Arduino IDE

IDE atau *Integrated Development Environment* adalah sebuah perangkat lunak yang digunakan untuk melakukan pengembangan pada Arduino. Arduino IDE dibuat dari bahasa *Java* yang kemudian dilengkapi dengan *library* C dan C++. Melalui perangkat lunak ini, pada Arduino dilakukan pemrograman untuk melakukan fungsi-fungsi yang dibenamkan melalui sintaks pemrograman. Pemrograman yang ada pada Arduino menyerupai bahasa C yang disebut *Sketch*.

Gambar 2.5 Tampilan perangkat lunak Arduino IDE

Bagian-bagian dari perangkat lunak Arduino IDE, antara lain :

- Verify

Bagian ini berfungsi untuk melakukan *checking code* yang dibuat apakah sudah sesuai dengan aturan pemrograman atau belum.

- Upload

Bagian ini berfungsi untuk melakukan kompilasi program atau *code* ke *board* Arduino.

- New

Bagian ini berfungsi untuk membuat *Sketch* baru.

- Open

Bagian ini berfungsi untuk membuka *sketch* yang pernah dibuat sebelumnya.

- Save

Bagian ini berfungsi untuk menyimpan *sketch* yang telah dibuat.

- Serial Monitor

Bagian ini berfungsi untuk menampilan program *debbuging* tanpa menggunakan LCD pada *board* Arduino. *Serial monitor* ini dapat digunakan untuk menampilkan nilai hasil pemrosesan, nilai hasil pembacaan sensor, bahkan memunculkan pesan *error.*

2.2.5 Modul *Wireless* NRF24L01

Gambar 2.6 Modul *Wireless* nRF24L01

(Sumber : dx.com)

NRF24L01 merupakan modul komunikasi serial nirkabel yang menggunakan *chip* original produksi Nordic Semiconductor dari Norwegia. NRF24L01 bekerja pada frekuensi ISM (Industrial, Scientific and Medical) 2,4GHz yang bebas lisensi dengan kecepatan hingga 2Mbps dengan pilihan opsi data rate 250Kbps, 1Mbps dan 2Mbps. Modul ini dilengkapi dengan tambahan *Power Amplifier* dan *Low Noise Amplifier* sehingga jarak transfer dapat semakin jauh dan lebih stabil. Jangkauan area modul ini mencapai 100m pada lapangan terbuka. Tegangan kerja dari modul ini adalah 5V DC. NRF24L01 memiliki *baseband logic Enhanced ShockBurst™ hardware protocol accelerator* yang s*upport "high-speed SPI interface for the application controller"*. NRF24L01 memiliki *true ULP solution*, yang memungkinkan daya tahan baterai berbulan-bulan hingga bertahun-tahun (Nordic Semiconductor ASA, 2016).

Modul ini memiliki 8 buah pin, diantaranya:

1. VCC (3.3V DC)
2. GND

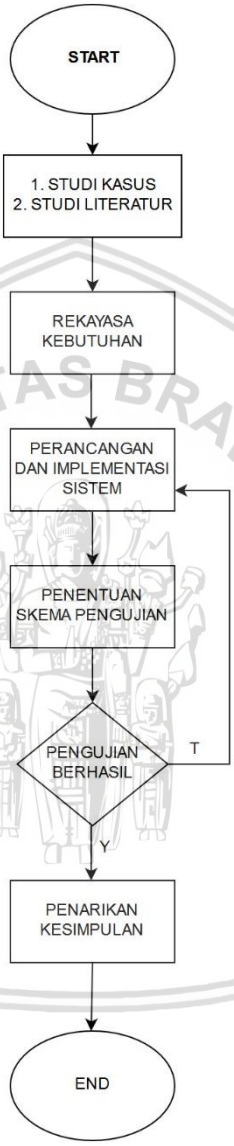
3. CE
4. CSN
5. MOSI
6. MISO
7. SCK
8. IRQ

Modul ini memiliki 126 pilihan channel sehingga memenuhi kebutuhan komunikasi-komunikasi multipoint dan frekuensi hopping yang dapat diatur oleh perangkat lunak. Komunikasi modul wireless nRF24L01 menggunakan interface SPI. Serial Peripheral Interface (SPI) merupakan salah satu jenis protokol komunikasi yang bersifat *full duplex*, yaitu ada device yang bertindak sebagai *Master* dan *Slave.* Bersifat *full duplex* karena *master* dan *slave* dapat mengirim atau menerima data dalam waktu yang bersamaan.

BAB 3 METODOLOGI

3.1 Alur Metode Penelitian

Alur metode penelitian secara umum dapat dilihat pada diagram alir seperti yang ditunjukkan pada Gambar 3.1 berikut ini.



Gambar 3.1 Diagram Alir Metode Penelitian

Pada alur metode penelitian setelah mengetahui masalah apa yang diangkat adalah menggali informasi dari literatur mengenai protokol *routing Ad-hoc On demand Distance Vector* (AODV). Selanjutnya dilakukan inisialisasi sistem pada perangkat keras. Lalu perencanaan pembuatan sistem manet yang meliputi: pembuatan dan konfigurasi *mobile node.* Selanjutnya pembuatan sistem pengiriman data yang meliputi: pembuatan dan konfigurasi protokol *routing* AODV. Langkah terakhir adalah pengujian dan analisis dari sistem yang telah dibuat. Hal ini akan dilihat juga apakah sistem yang diterapkan selama ini sesuai

dengan yang diharapkan atau tidak. Ketika hasil pengujian sudah selesai, maka dapat dilakukan pengambilan kesimpulan dari hasil yang didapatkan.

3.2 Studi Literatur

Pada bagian ini dibahas mengenai dasar teori yang mendukung penelitian implementasi *mobile ad-hoc network* (manet) dengan protokol AODV pada perangkat berbasis nRF24L01. Studi literatur yang dilakukan bertujuan untuk mengkaji hal-hal yang berhubungan dengan teori-teori yang mendukung dalam perencanaan dan perealisasian alat. Berikut merupakan dasar teori yang digunakan sebagai bahan studi:

1. Jaringan *Mobile Ad-Hoc Network*

   Pada bagian ini dilakukan kajian terkait dengan penelitian terdahulu yang telah menerapkan jaringan *Mobile Ad-Hoc Network* pada *Real Device*. Pada bagian ini juga dipelajari bagaimana tata cara penggunaan jaringan *Mobile Ad-Hoc Network* pada *Real Device*.

2. Protokol AODV

   Pada bagian ini dilakukan kajian terkait dengan penelitian terdahulu yang telah menerapkan protokol *routing* AODV, baik pada simulator dan *Real Device*. Pada bagian ini juga dipelajari mengenai tata cara penggunaan protokol AODV sebagai pengiriman data.

3. Modul Komunikasi *Wireless* nRF24L01

   Pada bagian ini dilakukan pencarian literatur terkait guna mengetahui cara penggunaan agar mampu diakses melalui mikrokontroler yang digunakan.

4. Mikrokontroler

   Pada bagian ini dilakukan pencarian literatur terkait guna mengetahui cara menggunakan atau mengakses mikrokontroler.

3.3 Analisis Kebutuhan Sistem

Analisis kebutuhan bertujuan untuk melihat semua kebutuhan yang selama ini diterapkan oleh sistem. Analisis kebutuhan dilakukan dengan mengindentifikasi kebutuhan sistem yang terdiri dari kebutuhan perangkat keras dan kebutuhan perangkat lunak. Dengan adanya pengidentifikasian ini nantinya akan mempermudah dalam melakukan pengujian dan analisis sistem.

3.3.1 Kebutuhan Perangkat Keras

Dalam penelitian ini menggunakan perangkat keras empat buah Arduino UNO dan empat buah Modul Wireless NRF24L01.

3.3.2 Kebutuhan Perangkat Lunak

Sedangkan perangkat lunak yang dibutuhkan untuk membuat perancangan, pengujian, serta memantau dan menganalisis adalah Arduino IDE.

3.4 Perancangan Penelitian

Tahap perancangan konsep sistem ini bertujuan agar perancangan sistem penelitian yang dilakukan menjadi terstruktur. Perancangan sistem penelitian dapat dijelaskan sebagai berikut:

1. Perancangan sistem dibagi menjadi 2 bagian, yaitu perancangan node yang meliputi subsistem node *base* dan node *client*.

2. Setiap node yaitu node *base* dan node *client* akan saling terhubung secara *mobile ad-hoc network (manet)*.

3. Node *base* akan menjadi tempat untuk *user* melakukan instruksi program dimana node *client* akan mengikuti instruksi program dari input *user* di node *base*.

4. Setelah semua node sudah saling terhubung, maka akan dijalankan instruksi untuk menjalankan protokol AODV untuk pengiriman pesan yang diinput oleh *user*.

5. Semua data yang telah dikirim dan diterima oleh tiap node, akan ditampilkan pada *serial monitor.*

6. Lalu akan diuji fungsionalitas sistem dengan skenario pengujian pengiriman data jika setiap node statis tanpa hambatan, bergerak dinamis tanpa hambatan, dan bergerak dinamis dengan adanya hambatan.

3.5 Implementasi

Implementasi meliputi proses perancangan sistem penelitian sampai dengan pada hasil akhir penelitian. Adapun tahapan dari implementasi sistem pada penelitian ini, yaitu:

1. Implementasi perangkat keras Arduino UNO dan modul komunikasi *wireless* nRF24L01 yang dapat dikontrol sekaligus dimonitor oleh komputer.

2. Implementasi *mobile ad-hoc network* pada tiap node, baik node *base* maupun node *client.*

3. Implementasi protokol AODV pada perangkat keras, beserta proses pengiriman data sesuai input dari *user*.

Dari proses implementasi tersebut diharapkan mendapatkan hasil sebagai berikut:

1. Perangkat Arduino UNO yang terhubung dengan modul komunikasi *wireless* nRF24L01 dapat dikontrol dan dimonitor oleh komputer.

2. Dapat mengimplementasikan jaringan *Mobile Ad-Hoc Network* (manet) menggunakan modul komunikasi *wireless* nRF24L01.

3. Keberhasilan pengiriman data dengan menggunakan protokol AODV pada modul komunikasi *wireless* nRF24L01.

4. Fungsionalitas sistem tetap berjalan dengan atau tanpa hambatan.

3.6 Pengujian dan Analisis

Pengujian pada skripsi ini dilakukan agar mengetahui bahwa sistem yang telah diterapkan sebelumnya, bekerja dengan baik sesuai dengan latar belakang dan spesifikasi yang melandasinya.

Analisis dilakukan setelah pengujian selesai. Jika dalam pengujian dan analisis telah didapatkan data yang cukup, maka selanjutnya akan dilakukan proses pengambilan kesimpulan.

3.7 Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan pengujian dan analisis sistem telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang telah berjalan pada penelitian ini.

3.8 Pembuatan Laporan Skripsi

Tahap ini merupakan tahap terakhir dari penelitian. Dilakukan penyusunan karya tulis penelitian sebagai dokumentasi dari pelaksanaan skripsi. Dokumentasi ini dibuat sebagai penjelasan dari hasil penelitian agar dapat mempermudah pengembangan penelitian selanjutnya.

BAB 4 REKAYASA KEBUTUHAN

Bab ini menjelaskan mengenai kebutuhan-kebutuhan sistem yang harus terpenuhi supaya dapat berjalan dengan baik dan berfungsi sesuai dengan tujuannya. Selain itu, rekayasa kebutuhan juga dijadikan sebagai dokumentasi untuk para pengembang sistem. Rekayasa kebutuhan ini meliputi pendahuluan, deskripsi umum dan analisis kebutuhan.

4.1 Pendahuluan

4.1.1 Tujuan

Tujuan dari pembuatan sistem adalah untuk melakukan implementasi *Mobile Ad-Hoc Network* dengan menggunakan protokol AODV pada modul *wireless* nRF24L01. Hasil implementasi ini diharapkan akan mempermudah para perancang *wireless sensor network* dalam menggunakan protokol AODV ke depannya. *Routing* protokol AODV ini hanya menyimpan info *routing* seputar *path* dan *host* yang aktif. Setiap *node* menyimpan tabel *routing next-hop*, artinya dia menyimpan info tujuan *hop* berikutnya. Jadi, ketika *node base* ingin mengirimkan paket ke tujuan namun tak ada *route* tersedia, *node base* akan memulai proses *Route Discovery*, yakni *Node base* akan *broadcast Route Request* (RREQ) paket yang disertai nomor *sequence* tujuan. Ketika *Node* tujuan atau *node* yang memiliki *route* menuju ke tujuan menerima paket RREQ, *node* tersebut akan memeriksa nomor *sequence* tujuan yang sampai di *node* tersebut ketika paket tiba apakah nomor *sequence* sama dengan info dari RREQ. Untuk memastikan kalau paket tersebut masih baru, *node* tujuan membalas paket RREQ dengan *Route Reply* (RREP) paket. RREP dibuat dan dikirim lagi ke *node base* hanya jika nomor *sequence* sama atau lebih besar dari yang dispesifikasikan oleh RREQ. AODV hanya menggunakan Link yang bersifat simetris dan RREP mengikuti *path* sebaliknya dari yang dihasilkan RREQ. Ketika menerima RREP, setiap *node* antara *base* dan tujuan mengupdate *routing table next-hop* dengan RREP tujuannya. *Node base* kemudian memilih *route* paling sedikit untuk mengirim paket tujuannya.

4.1.2 Ruang Lingkup

Sistem ini terdiri dari empat *node* yaitu satu *node base* sebagai *input* intruksi dari *User*, menampilkan *output* hasil *table routing* dan data yang dikirim, sementara tiga *node* lainnya sebagai *tranceiver*.

4.1.3 Definisi, Istilah dan Singkatan

Definisi, istilah dan singkatan yang terdapat pada dokumen penelitian ini meliputi:

1. *Transceiver* : komponen elektronik yang memadukan komponen *transmitter* dan *receiver*.
2. *Transmitter* : Pengirim paket data
3. *Receiver* : Penerima paket data

4. *Node* : sebuah jaringan komputer yang terdiri dari dua unit komputer atau lebih yang saling terhubung satu sama yang lain.

4.1.4 Sistematika

Sistematika merupakan kerangka dan pedoman penulisan dokumentasi. Sistematika pada rekayasa kebutuhan sistem ini terbagi menjadi 3 bagian:

1. Pendahuluan, berisi tentang tujuan pembuatan sistem, ruang lingkup, istilah dan sistematika dari sistem.

2. Deskripsi umum, berisi penjelasan secara umum mengenai sistem yang akan dibuat meliputi perspektif, kegunaan, karakteristik pengguna, lingkungan operasi dan batasan perancangan dan implementasi dari sistem.

3. Analisis Kebutuhan, berisi penjelasan mengenai kebutuhan fungsional, perangkat keras dan perangkat lunak.

4.2 Deskripsi Umum

Deskripsi umum menjelaskan secara rinci mengenai kebutuhan yang harus dipenuhi untuk perancangan dan implementasi sistem. Penjelasan deskripsi umum meliputi perspektif sistem, kegunaan, karakteristik sistem, lingkungan operasi dan batasan perancangan dan implementasi pada sistem.

4.2.1 Perspektif Sistem

Sistem dapat dikatakan berhasil apabila dapat berhasil mengimplementasikan protokol AODV dan mengirim data menggunakan modul *wireless* nRF24L01. Parameter uji yang telah ditentukan yaitu dapat mengirim dan menerima RREQ dan RREP, menampilkan tabel *routing* dan mengirim data. Tahap awal, sistem akan mulai melakukan proses *Route Discovery*, yakni *Node base* akan *broadcast Route Request* (RREQ) paket yang disertai nomor *sequence* tujuan ke seluruh *node* terdekat. Kemudian, *Node* tujuan atau *node* terdekat yang memiliki *route* menuju ke tujuan menerima paket RREQ, *node* tersebut akan memeriksa nomor *sequence* tujuan yang sampai di *node* tersebut ketika paket tiba apakah nomor *sequence* sama dengan info dari RREQ. Untuk memastikan kalau paket tersebut masih baru, node tujuan membalas paket RREQ dengan *Route Reply* (RREP) paket. RREP dibuat dan dikirim lagi ke *node base*. Paket tersebut akan terus dikirim hingga semua *node* mengirimkan RREP ke *node base*. Waktu pengiriman RREQ dan RREP memiliki interval 15 detik. *Node base* akan menerima data dari semua *node* dan akan ditampilkan di serial monitor. *Node base* juga akan memberikan input data jika semua proses dan *routing table* sudah terbentuk yang akan ditampilkan pada serial monitor juga. Selain itu, pengujian fungsionalitas sistem juga dikatakan berhasil dilakukan apabila setiap node diam statis, node bergerak dinamis tanpa hambatan, dan node bergerak dinamis dengan hambatan sukses mengirimkan paket data menuju node tujuan.

4.2.2 Karakteristik Pengguna

Sistem akan diuji oleh penulis selaku pembuat sistem.

4.2.3 Lingkungan Operasi

Pengujian sistem ini akan dilakukan ditempat tertutup (*indoor*) dan terbuka (outdoor), dengan dan tanpa penghalang.

4.2.4 Batasan Perancangan dan Implementasi

Berikut adalah batasan perancangan dan implementasi sistem yang telah ditentukan.

1. Sebagai *input* data oleh *user* untuk menguji kinerja sistem, maka digunakan satu *node base*. Tiga *node* lainnya akan berfungsi sebagai *transceiver* dan *transmitter*.

2. Kinerja yang diuji adalah mengirimkan paket RREQ dan RREP, menampilkan *table routing*, dan mengirimkan data.

3. Pengujian sistem dilakukan di dalam ruangan (*indoor*) tanpa penghalang

4. Pengujian sistem dilakukan menggunakan modul *wireless* nRF24L01

4.3 Analisis Kebutuhan

Analisis kebutuhan menjelaskan berbagai macam kebutuhan yang digunakan untuk merancang sistem. Sub bab ini meliputi kebutuhan fungsional sistem, kebutuhan perangkat keras, dan kebutuhan perangkat lunak.

4.3.1 Kebutuhan Fungsional

Adapun kebutuhan fungsional yang harus dipenuhi dalam penelitian ini adalah:

1. *Node base* mengirimkan paket *discovery* secara *broadcast* ke setiap *node client*.

Node base merupakan *node* yang mengirimkan paket *discovery* harus mampu mengirimkan paket tersebut kepada setiap *node Client*. Pengiriman paket *discovery* ini sebagai tahap awal dimulainya proses *routing* yang dilakukan oleh *node base*.

2. *Node client* menerima paket *discovery* dari *node base*

Paket discovery yang diterima oleh *node client* dari *node base* berisikan:

- Jenis paket (Q) menunjukkan bahwa itu paket RREQ

- ID pengirim paket

- ID tujuan paket

- Pesan *Discovery*. Disini penulis sisipkan pesan “hello” di paket *discovery* tersebut

- *Next-Hop*

3. *Node base* dan *node client* saling bertukar paket data

Pada tahap ini semua *node* akan mulai bertukar paket data antara *node* satu dengan yang lain. Setelah paket *discovery* tersebar ke setiap *node client*, maka *node client* akan dapat melakukan pertukaran paket data dengan *node base* dan

node-node yang lain. *Node client* akan mengirimkan paket *request reply* (RREP) menuju *node base* melalui *path* sebaliknya yang terbentuk setelah RREQ berhasil.

4. *Node client* mampu menampilkan data di serial monitor yang berisi pesan:

- *Error*. Apabila paket RREQ gagal diterima dan node client menunggu dalam interval 15 detik untuk dikirimkan paket ulang.

- *Broadcast*. Apabila paket RREQ yang diterima sampai di *node* yang bukan tujuannya. Maka *node* yang menerima paket RREQ ini akan meneruskan menuju *node* tujuan

- *Got* RREQ, *Sending* RREP. Apabila paket RREQ berhasil sampai di *node* tujuan, maka *node* tujuan membuat paket RREP untuk dikirimkan balik ke *Node base* melalui path yang sama.

5. *Node base* mampu menerima paket data yang dikirimkan oleh *node client* dan menampilkan datanya dalam serial monitor *node base*.

Node base bertindak sebagai *receiver* yang akan menampung data paket RREP yang telah dilakukan oleh *node client*. Kemudian akan ditampilkan dalam serial monitor node base. Paket tersebut berisikan:

- Jenis paket (P) menunjukkan bahwa itu paket RREP

- ID pengirim paket

- ID tujuan paket

- Pesan *Reply*. Disini penulis sisipkan pesan "hello" di paket RREP tersebut

- *Next-Hop*

4.3.2 Kebutuhan Non Fungsional

Kebutuhan non fungsional rnenjelaskan mengenai apa saja penunjang dari kebutuhan fungsional sistem. Adapun kebutuhan non fungsional yang harus dipenuhi dalam penelitian ini adalah:

1. Kebutuhan daya yang digunakan

Setiap komponen membutuhkan daya yang sesuai agar dapat bekerja dengan baik. nRF24L01 membutuhkan daya sebesar 3.3V. *Board* UNO sendiri membutuhkan daya sebesar 5V juga agar dapat memenuhi kebutuhan daya pada komponen yang terhubung dengan Arduino

2. Penggunaan *channel* pada nRF24L01

Agar *node client* dan *node base* dapat saling berkomunikasi menggunakan modul komunikasi *wireless* nRF24L01, maka *channel* atau saluran yang digunakan antar modul komunikasi dalam tiap *node* harus diatur dalam satu *channel* yang sama. pada penelitian ini *channel* yang digunakan adalah 99.

4.3.1 Perangkat Keras

Adapun kebutuhan perangkat keras yang digunakan dalam proses implementasi sistem, antara Iain:

1. Komputer/ Laptop

Komputer digunakan untuk memonitor sekaligus memberikan *input-output* yang dibutuhkan, yaitu *input* program untuk mikrokontroler dan *output* mikrokontroler yang ditampilkan dalam serial monitor.

2. Arduino UNO

Arduino UNO merupakan papan mikrokontroler yang akan ditanami program sehingga dapat bekerja sesuai dengan topik penelitian. Arduino UNO digunakan juga sebagai perangkai Pengolah data yang akan dikirimkan maupun diterima.

3. Modul *Wireless* nRF24L01

nRF24L01 merupakan salah satu jenis modul *wireless* yang digunakan untuk melakukan komunikasi data pada penelitian ini.

4. Kabel *Jumper*

Kabel *jumper* dibutuhkan untuk menghubungkan antara Modul *Wireless* nRF24L01 dengan Arduino UNO.

5. PCB

PCB disini dibutuhkan sebagai alternatif jika slot kabel *jumper* yang ada di Arduino UNO, yang seharusnya menghubungkan antara Modul Wireless nRF24L01 dengan Arduino UNO longgar, mengakibatkan perangkat tidak terhubung dengan benar, maka PCB digunakan untuk menghubungkan langsung antara Modul *Wireless* nRF24L01 dengan Arduino UNO.

4.3.2 Perangkat Lunak

Adapun kebutuhan perangkat lunak dalam penelitian ini, meliputi:

1. Microsoft Windows 10 Pro 64-bit

Perangkat lunak ini digunakan sebagai sistem operasi yang digunakan oleh komputer/ laptop. Dalam hal ini sistem operasi jenis lain juga dapat digunakan asalkan kompatibel dengan Arduino IDE.

2. ArduinolDE

Perangkat lunak ini digunakan untuk menuliskan program meng*compile* program dan meng*upload* program ke mikrokontroler. Perangkat lunak ini juga membantu peneliti untuk melakukan melihat data hasil olahan sekaligus memonitor jalannya program dengan menggunakan menu serial monitor didalamnya.

4.4 Batasan Desain Sistem

Dalam proses implementasi, sistem ini memiliki beberapa keterbatasan oleh karena spesifikasi perangkat keras yang digunakan. Seperti pada modul komunikasi *wireless* nRF24L01 memiliki keterbatasan pada jarak transmisi yang tidak terlalu jauh. Selain itu dalam penggunaan nRF24L01 dibutuhkan lingkungan yang bebas interferensi gelombang dan juga daya yang diperlukan oleh sistem ini harus tepat pada 5V (Ball, 2007). Agar pengiriman data berbasis *wireless* ini dapat dilakukan sesuai dengan harapan, maka perlu diterapkan batasan-batasan implementasi desain sistem, antara lain :

1. Sistem menggunakan mikrokontroler Arduino UNO.
2. Sistem dalam proses komunikasinya menggunakan modul *wireless* nRF24L01.
3. Implementasi dilakukan dengan membuat perangkat *node base* dan *node client*
4. *Node client* terdiri dari 3 *node* yang masing-masing mengirimkan paket data menuju *node base*.
5. Implementasi antar *node* (*node client* dan *node base*) menggunakan protokol AODV.
6. *Output* dari data yang diterima oleh *node base* akan ditampilkan pada serial monitor *node base*. *Output* yang ditampilkan berupa *header* paket, alamat tiap node pengirim dan tujuan, dan pesan yg dikirimkan.
7. Peletakan *node client* berada pada *range* deteksi tidak lebih dari 10 meter.
8. *Input* dari implementasi sistem adalah dari keyboard pada *node base*.
9. Hambatan yang digunakan adalah pintu dan dinding.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab ini menguraikan proses implementasi dari dasar teori yang telah dipelajari sesuai analisis dan perancangan sistem.

5.1 Perancangan

Perancangan penelitian ini dilakukan untuk merencanakan tahapan yang harus dilakukan dalam melakukan penelitian. Tahapan penelitian yang harus dilakukan antara lain identifikasi kebutuhan yang akan digunakan dalam penelitian, baik pada sisi perangkat keras maupun perangkat lunak, serta perancangan proses implementasi. Diharapkan dengan dillakukannya tahapan perancangan dapat mempermudah pelaksanaan penelitian.

5.1.1 Pendefinisian dan Perancangan Penelitian

Penelitian dilakukan dengan menerapkan protokol AODV yang diimplementasikan pada Arduino UNO. Implementasi protokol AODV pada perangkat nRF24L01 yang berbasis *wireless* ini dilakukan dengan merangkai seluruh perangkat keras yang dibutuhkan agar dapat bekerja sekaligus dapat berkomunikasi sesuai dengan kebutuhan penelitian. Perangkat keras yang digunakan meliputi Arduino UNO dan modul komunikasi *wireless* nRF24L01. Protokol AODV diterapkan pada mikrokontroler *Node base* agar dapat berkomunikasi terhadap *node client*. Proses sebelumnya yaitu dilakukannya *broadcast* antar *node*, baik *node client* maupun *node base*. Kemudian dilakukan pengenalan posisi di *routing table*. Setelah itu dikirimkanlah data *string* oleh masing-masing *node client* ke *node base* dengan menggunakan modul komunikasi *wireless* nRF24L01. Hasil tersebut akan ditampilkan pada serial monitor. Selain itu, akan dilakukan pula pengujian fungsionalitas sistem, dimana skenario yang digunakan adalah percobaan mengirim data dengan posisi *node* statis tanpa hambatan, *node* dinamis tanpa hambatan, dan *node* dinamis dengan hambatan.

5.1.1.1 Perancangan Perangkat Keras

Perancangan perangkat keras didasarkan pada analisis kebutuhan penelitian yang telah ditetapkan di awal penelitian. Terdapat 2 komponen perangkat keras yang digunakan, yaitu: Arduino UNO dan modul wireless nRF24L01. Untuk mendeteksi nRF24L01 di sekitarnya, masing-masing modul mengirimkan pesan *broadcast* yang akan diterima dan akan disimpan dalam *routing table*. Setelah itu dikirimkanlah data *string* oleh masing-masing *node client* ke *node base* dengan menggunakan modul komunikasi *wireless* nRF24L01. Hasil tersebut akan ditampilkan pada serial monitor.

5.1.1.2 Perancangan Subsistem *Node Base*

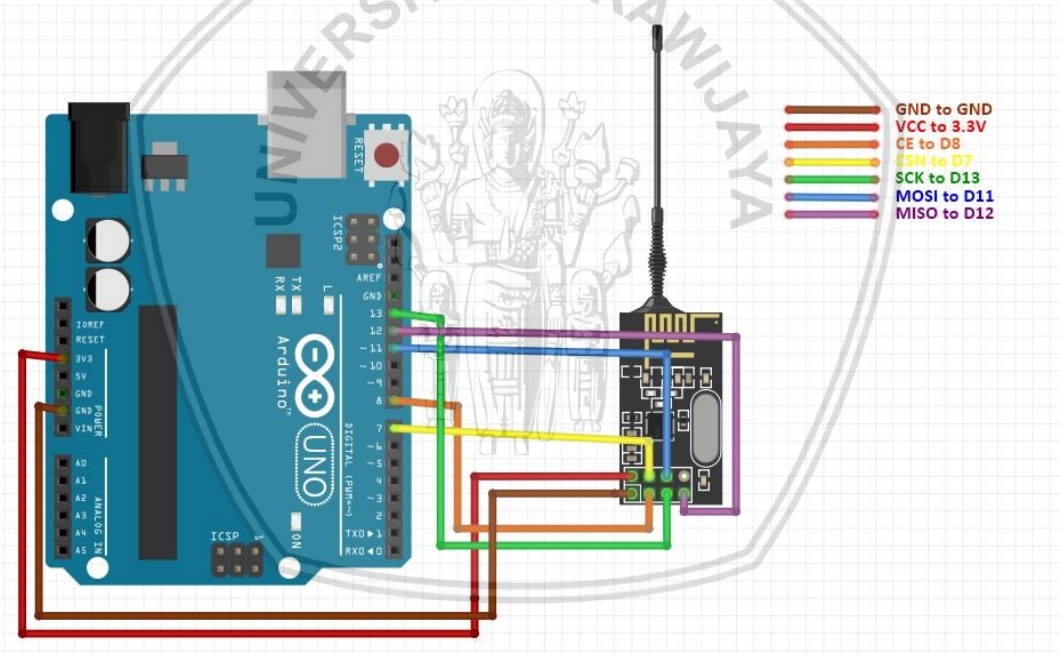
Gambar 5.1 Diagram Sistem *Node Base*

Penjelasan diagram sistem pada *node base* di Gambar 5.1:

1. *Power Supply* merupakan daya yang dibutuhkan *node* untuk menjalankan fungsinya. Pada penelitian ini, *power supply* berupa tegangan sebesar 3.3V yang didapatkan dari Port komputer/Laptop/*Powerbank* yang digunakan.

2. Mikrokontroler pada *node base* menggunakan Arduino UNO bertindak sebagai pengirim data dan sebagai kontrol komunikasi dengan *receiver node client*.

3. nRF24L01 merupakan jenis modul *wireless* yang digunakan pada penelitian ini yang bertindak sebagai media komunikasi data.



Gambar 5.2 Perancangan Sistem *Node Base*

Pada penelitian ini, perancangan sistem *node base* yaitu dengan menghubungkan mikrokontroler Arduino UNO dengan modul komunikasi *wireless* nRF24L01 seperti dapat dilihat pada Gambar 5.2. Modul komunikasi *wireless* pada *node base* digunakan untuk melakukan komunikasi dengan *node client*. Perancangan sistem dilakukan dengan menghubungkan pin-pin yang ada pada nRF24L01 dengan pin yang ada pada *board* Arduino UNO. Bagaimana modul *wireless* nRF24L01 dan Arduino UNO terhubung dapat dilihat pada Tabel 5.1

Tabel 5.1 Konfigurasi Pin Arduino UNO dan nRF24L01 *Node Base*

nRF24L01	Warna Kabel	Arduino UNO
Nama PIN		Nama PIN
GND	Cokelat	GND
VCC	Merah	3.3V
CE	Oranye	D8
CSN	Kuning	D7
SCK	Hijau	D13
MOSI	Biru	D11
MISO	Ungu	D12

5.1.1.3 Perancangan Subsistem *Node Client*

Gambar 5.3 Diagram Sistem *Node Client*

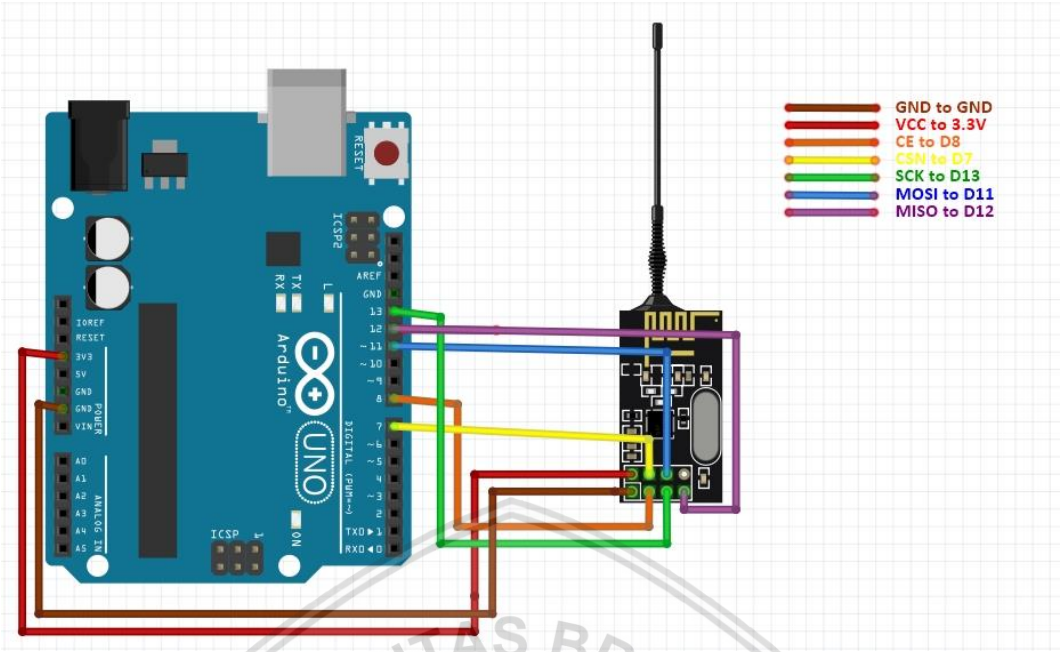
Penjelasan diagram sistem pada *node Client* di Gambar 5.3:

1. *Power Supply* merupakan daya yang dibutuhkan *node* untuk menjalankan fungsinya. Pada penelitian ini, *power supply* berupa tegangan sebesar 3.3V yang didapatkan dan *port* USB komputer/Laptop/*Powerbank* yang digunakan.

2. Mikrokontroler pada *node client* menggunakan Arduino UNO yang bertindak sebagai penerima data dan sebagai *tranceiver node base*.

3. nRF24L01 merupakan jenis modul *wireless* yang digunakan pada penelitian ini yang bertindak sebagai media komunikasi data.

Pada penelitian ini, perancangan sistem *node client* sama saja dengan *node base* yaitu dengan menghubungkan mikrokontroler Arduino UNO dengan modul komunikasi *wireless* nRF24L01 seperti dapat dilihat pada Gambar 5.4. Modul komunikasi *wireless* pada *node client* digunakan untuk melakukan komunikasi dengan *node base*. Perancangan sistem dilakukan dengan menghubungkan pin-pin yang ada pada nRF24L01 dengan pin yang ada pada board Arduino UNO. Bagaimana modul *wireless* nRF24L01 dan Arduino UNO terhubung dapat dilihat pada Tabel 5.2

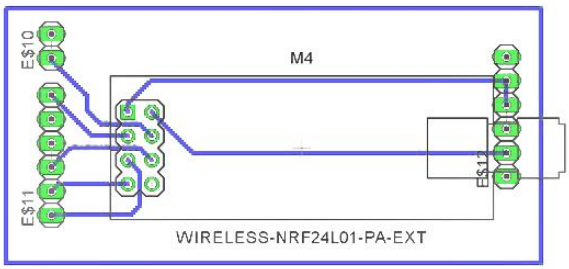


Gambar 5.4 Perancangan Sistem *Node Client*

Tabel 5.2 Konfigurasi Pin Arduino UNO dan nRF24L01 *Node Client*

nRF24L01	Warna Kabel	Arduino UNO
Nama PIN		Nama PIN
GND	Cokelat	GND
VCC	Merah	3.3V
CE	Oranye	D8
CSN	Kuning	D7
SCK	Hijau	D13
MOSI	Biru	D11
MISO	Ungu	D12

5.1.1.4 Perancangan Penggunaan *PCB*



Gambar 5.5 Perancangan Penggunaan PCB

PCB disini dibutuhkan sebagai alternatif jika slot kabel jumper yang ada di Arduino UNO, yang seharusnya menghubungkan antara Modul *Wireless* nRF24L01

dengan Arduino UNO longgar, mengakibatkan perangkat tidak terhubung dengan benar, maka PCB digunakan untuk menghubungkan secara langsung antara Modul *Wireless* nRF24L01 dengan Arduino UNO. Rancangan desain PCB bisa dilihat di Gambar 5.5.

5.1.2 Perancangan Komunikasi Perangkat Keras

Komunikasi antara *node client* dan *node base* pada penelitian ini menggunakan komunikasi *wireless* dengan menggunakan modul *wireless* nRF24L01. Pada penelitian ini menggunakan komunikasi data serial, dimana data akan dikirimkan per-*bit* data. Pengiriman data secara *Wireless* pada prinsipnya menumpangkan data pada pita frekuensi pembawa. Frekuensi yang digunakan untuk melewatkan data adalah pada 2,4GHz global ISM Band.

Frekuensi dan *channel* yang digunakan dalam penelitian ini diatur secara otomatis. Modul komunikasi *wireless* nRF24L01 menggunakan *range channel* antara 0-125. Penggunaan *channel* juga mempengaruhl frekuensi yang digunakan oleh tiap *node* dalam jaringan. Penggunaan frekuensi pada modul komunikasi *wireless* nRF24L01 ini dimulai dari 2.400GHz sampai dengan 2.525GHZ. Pada data *sheet* nRF24L01 terdapat rumus untuk menentukan frekuensi yang digunakan, yaitu:

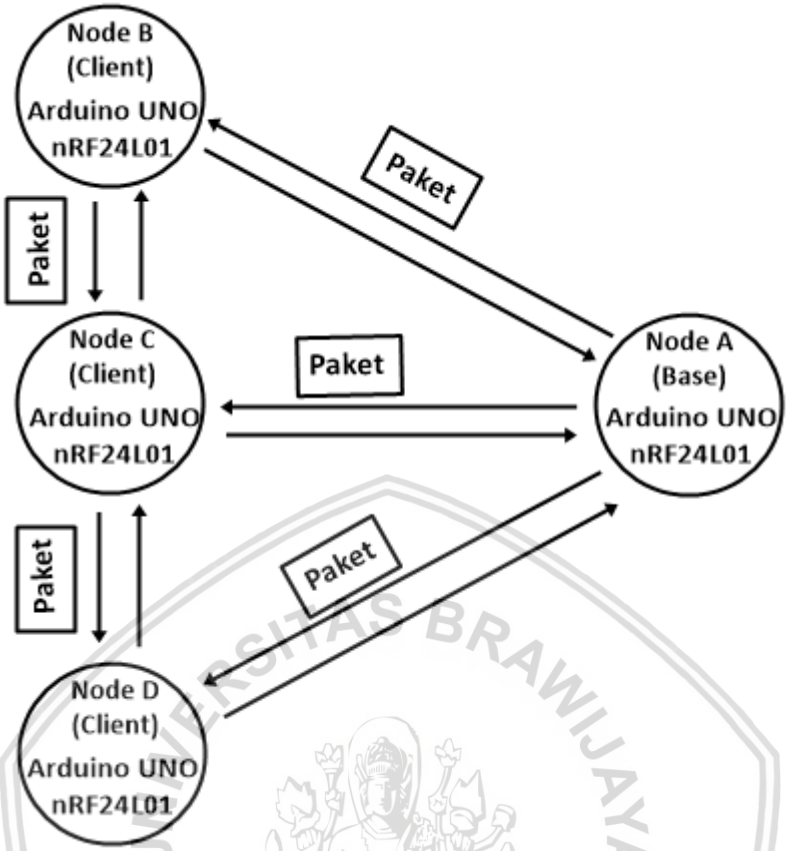
$$F0 = 2400 + RF\ Channel\ [MHz]$$

Pada penelitian ini menggunakan *channel* 99, maka frekuensi yang digunakan adalah:

$$F0 = 2400 + 99$$
$$F0 = 2499\ MHz = \mathbf{2.5\ GHz}$$

5.1.3 Perancangan Protokol AODV

Routing protokol AODV ini hanya menyimpan info *routing* seputar *path* dan *host* yang aktif. Setiap *node* menyimpan tabel *routing next-hop*, artinya dia menyimpan info tujuan *hop* berikutnya. Jadi, ketika *node base* ingin mengirimkan paket ke tujuan namun tak ada *route* tersedia, *node base* akan memulai proses *Route Discovery*, yakni *Node base* akan *broadcast Route Request* (RREQ) paket yang disertai nomor *sequence tujuan*. Ketika *Node* tujuan atau *node* yang memiliki *route* menuju ke tujuan menerima paket RREQ, *node* tersebut akan memeriksa nomor *sequence* tujuan yang sampai di *node* tersebut ketika paket tiba apakah nomor *sequence* sama dengan info dari RREQ.

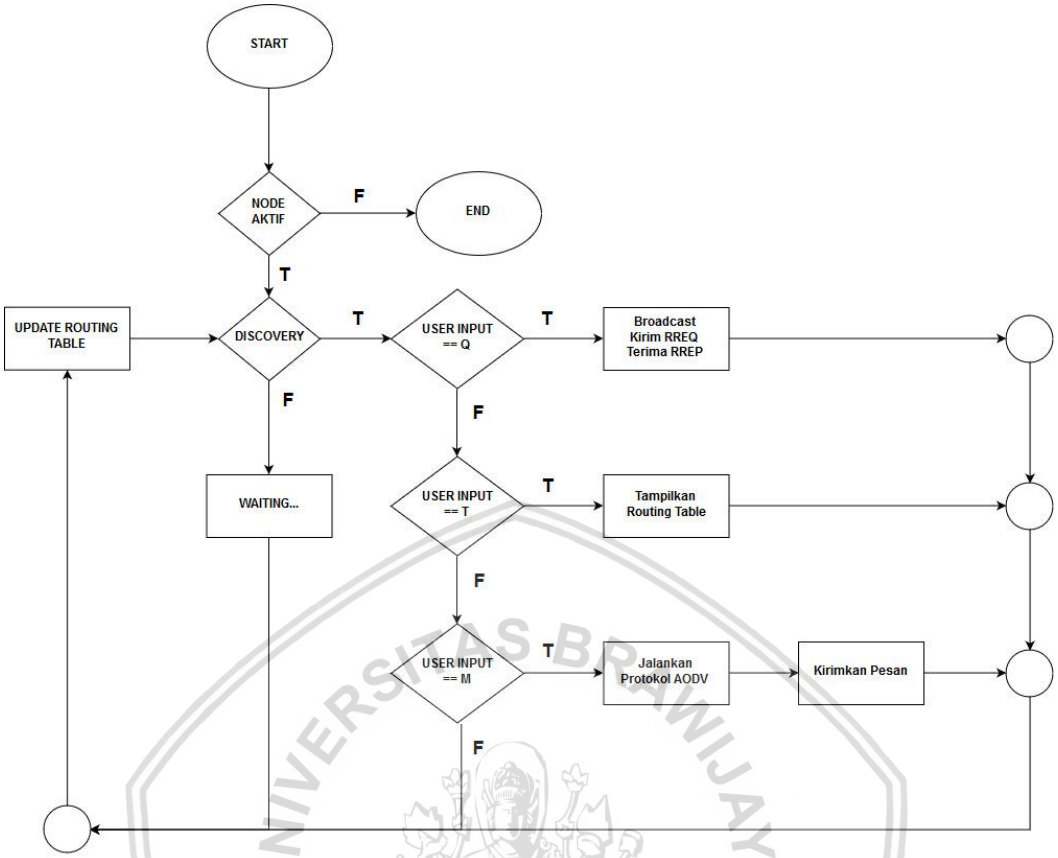


Gambar 5.6 Perancangan Protokol AODV

Untuk memastikan kalau paket tersebut masih baru, *node* tujuan membalas paket RREQ dengan *Route Reply* (RREP) paket. RREP dibuat dan dikirim lagi ke *node base* hanya jika nomor *sequence* sama atau lebih besar dari yang dispesifikasikan oleh RREQ. AODV hanya menggunakan *Link* yang bersifat simetris dan RREP mengikuti *path* sebaliknya dari yang dihasilkan RREQ. Ketika menerima RREP, setiap *node* antara *base* dan tujuan mengupdate *routing table next-hop* dengan RREP tujuannya. *Node base* kemudian memilih *route* paling sedikit untuk mengirim paket tujuannya seperti pada Gambar 5.6. Pada penelitian ini ada jeda waktu selama 15 detik untuk mengirimkan paket data pertama ke paket data selanjutnya.

5.1.4 Perancangan Pengiriman Data *Node Base*

Node Base berfungsi sebagai pengirim data atau *transmitter* yang bertugas untuk mengirim data *string*. Alur kerja pengiriman data *node base* dapat dilihat pada Gambar 5.7



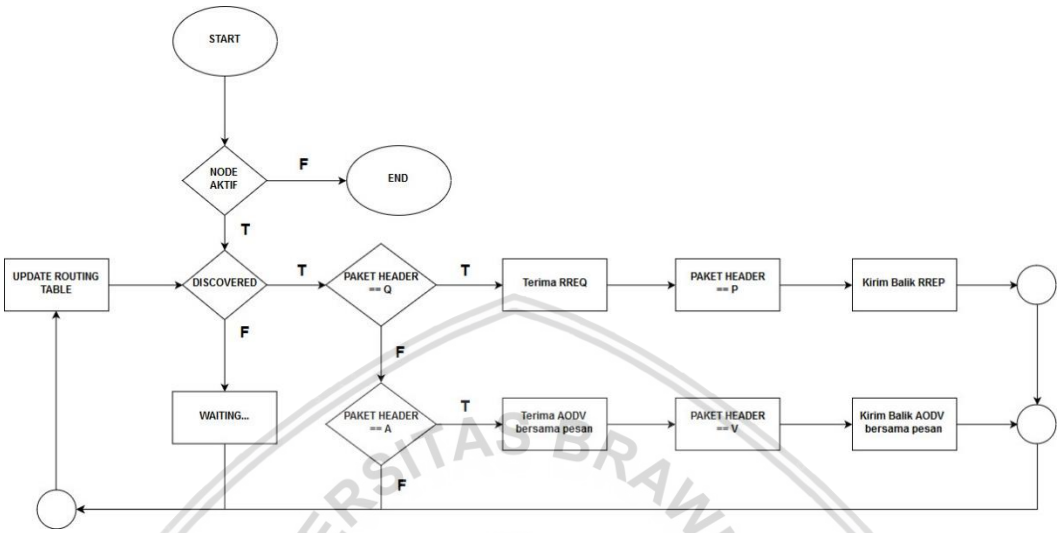
Gambar 5.7 Diagram Alur Pengiriman Data *node base*

Proses pengiriman data pada sisi *node base* diawali dengan melakukan proses inisialisasi. Proses inisialisasi ini meliputi inisialisasi *baudrate*, pin *client*, dan *channel*. Inisialisasi *baudrate* berfungsi untuk menentukan mikrokontroler yang digunakan bekerja pada *rate* yang telah ditentukan. Inisialisasi pin berfungsi untuk menentukan *node* bekerja pada pin yang telah ditetapkan. Inisialisasi *channel* berfungsi untuk menentukan *channel* komunikasi *wireless* nRF24L01 antara *node base* dan *node client*.

Setelah *node base* aktif maka *node* akan melakukan proses pengenalan. Pada proses ini, ketika *node base* ingin mengirimkan paket ke tujuan namun tak ada *route* tersedia, *node base* akan memulai proses *Route Discovery*, yakni *Node base* akan *broadcast Route Request* (RREQ) paket yang disertai nomor *sequence* tujuan ke semua tetangganya. Ketika *Node* Tetangga yang menerima paket permintaan tersebut atau *node* yang memiliki *route* menuju ke tujuan menerima paket RREQ, *node* tersebut akan memeriksa nomor *sequence* tujuan yang sampai di *node* tersebut ketika paket tiba apakah nomor *sequence* sama dengan info dari RREQ. Ketika salah satu *node client* telah berhasil terhubung dengan *node base*, maka *node* tersebut akan menjadi meneruskan kembali paket *discovery* tersebut ke *node* tetangganya, hingga semua *node* memiliki rute menuju *node* selanjutnya. Setelah semua *node* memiliki rute, barulah dijalankan *routing protokol* AODV yang disertai dengan pengiriman pesan dari *user*.

5.1.5 Perancangan Penerimaan Data *Node Client*

Node client merupakan sebuah *node* yang berfungsi sebagai *receiver* data yang dikirimkan oleh *node base* . Alur kerja *node* dapat dilihat pada Gambar 5.8.



Gambar 5.8 Diagram Alur Penerimaan Data *Node client*

Setelah *Node client* aktif maka *node* akan menunggu kiriman paket *discovery* selama maksimal 15 detik. Ketika sudah menemukan *node* tujuan, maka *node* tujuan akan mengirim *Route Reply* (RREP) menuju *node base* sebagai pesan *unicast*. *Hop* berikutnya ditentukan berdasarkan oleh alamat pengirim di table *routing* yang sudah terbentuk. Paket RREP hanya akan dikirimkan melalui jalur simetris sebaliknya yang sudah terbentuk melalui RREQ sebelumnya. Pengiriman paket itu terus berlanjut, hingga paket balasan itu mencapai node base. *Table Routing* juga sudah terbentuk dan ini menyelesaikan proses pencarian rute. sekarang karena rute dari *node base* ke *node* tujuan sudah ditentukan maka pesan dapat dikirimkan. Setelah *route* terbentuk, maka *node client* siap menerima paket *routing* protokol AODV yang disertai dengan pesan input dari *user*, dan mengirimkannya kembali ke *node base* untuk ditampilkan di serial monitor.

5.2 Implementasi

Pada tahap ini akan dibahas mengenai implementasi dari semua perancangan penelitian yang telah disusun sebelumnya. Pada tahap ini akan dijelaskan langkah-langkah penelitian dalam mengimplementasikan protokol AODV pada nRF24L01.

5.2.1 Spesifikasi Perangkat keras

Perancangan penelitian ini akan menggunakan komponen perangkat keras Arduino UNO dan modul *wireless* nRF24L01. Spesifikasi dari komponen-komponen tersebut adalah:

- Perangkat Mikrokontroler

Mikrokontroler yang digunakan dalam penelitian ini adalah Arduino UNO. Spesifikasi dari Arduino UNO dapat dilihat pada Tabel 5.3.

Tabel 5.3 Spesifikasi Arduino UNO

Arduino UNO	
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

- Modul *wireless* nRF24L01

Modul *wireless* nRF24L01 digunakan untuk mengirimkan data (komunikasi data) antar *node client* dan *node base*. Spesifikasi modul *wireless* nRF24L01 dapat dilihat pada Tabel 5.4.

Tabel 5.4 Spesifikasi Modul Wireless nRF24L01

Modul nRF24L01	
Frequency	2,4GHz ISM Band
Channels	126 RF
Data Rate	250Kbps, 1 and 2Mbps
Transmitter	11,3mA at 0dBm output power

Digital Interface	SPI Speed 0-8Mhz
Voltage	1,9 ~ 3,6V

- Laptop / Komputer PC

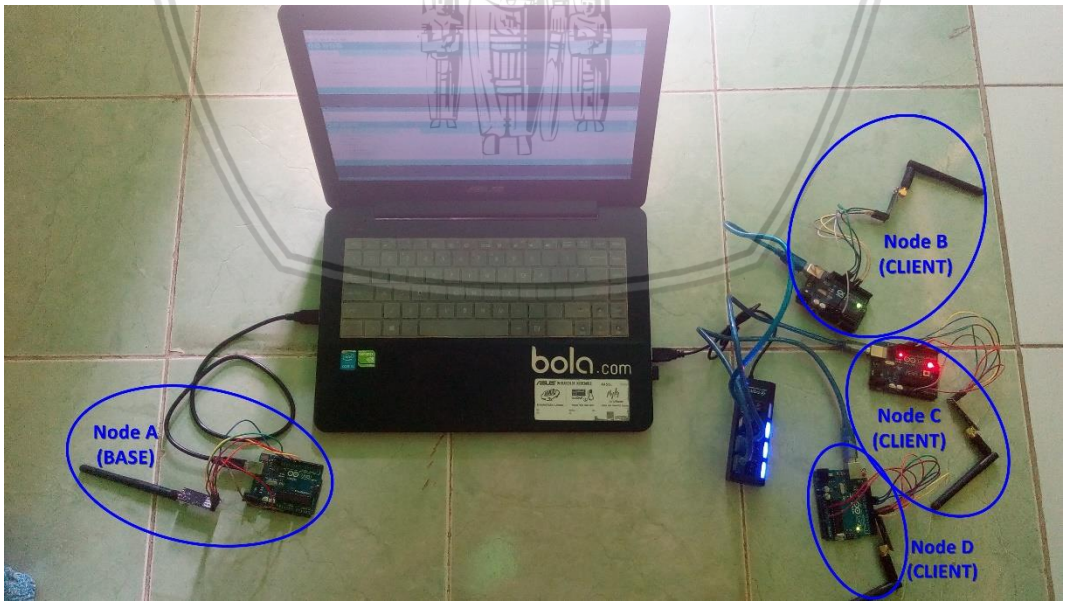
Laptop digunakan untuk membuat sekaligus meng*upload* program ke mikrokontroler. Spesifikasi laptop yang digunakan pada penelitian ini dapat dilihat pada Tabel 5.5.

Tabel 5.5 Spesifikasi Komputer/Laptop

Laptop ASUS X455LD	
Processors	Intel Core i3-4030U 1.9GHz
Memory (RAM)	4GB
Harddisk	500GB

5.2.2 Implementasi Perangkat Keras

Pada masing masing *node*, implementasi komunikasi perangkat keras dilakukan secara serial dengan menghubungkan *board* Arduino ke Laptop untuk pemrograman dan pengujian. Pada *node base* dan *node client*, untuk proses komunikasi atau pengiriman data akan dilakukan secara *wireless* dengan menggunakan modul *wireless* nRF24L01

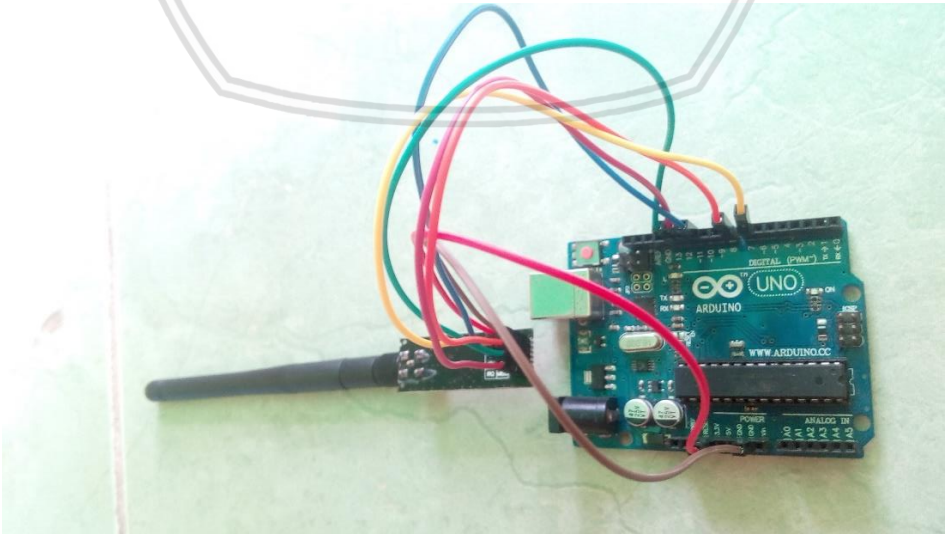


Gambar 5.9 Implementasi dengan kabel *jumper*

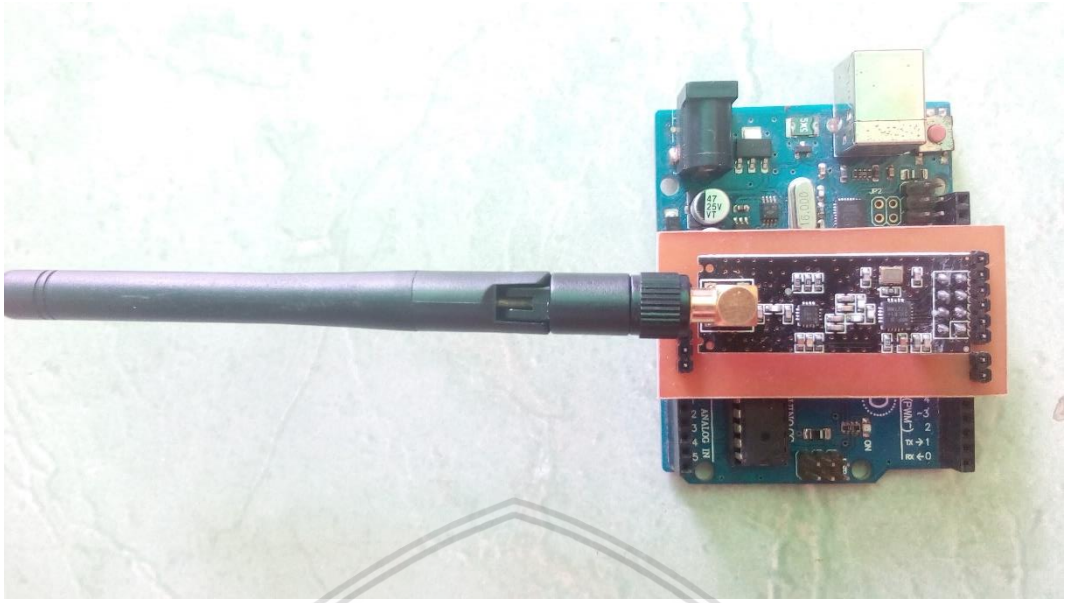
Gambar 5.10 Implementasi dengan PCB

Gambar 5.9 dan 5.10 merupakan implementasi komponen sistem berupa nRF24L01 dan Arduino UNO untuk melakukan komunikasi data *wireless* antar *node*.

Pada penelitian ini, implementasi perangkat keras adalah pemasangan seluruh komponen-komponen yang digunakan. Terdapat 2 pengelompokan perangkat keras, yaitu : perangkat keras pada *Node base* dan perangkat keras pada *node client*. Pemasangan perangkat keras pada *node base* meliputi Arduino UNO untuk menampilkan data hasil *input* dan *receiver* modul komunikasi *wireless* nRF24L01 yang digunakan untuk menerima data. Sedangkan pada *node Client* semua terhubung melalui tambahan USB Hub.



Gambar 5.11 Detail Komponen Perangkat keras menggunakan kabel *jumper*



Gambar 5.12 Detail Komponen Perangkat keras menggunakan PCB

Pada Gambar 5.11 dan 5.12 diatas merupakan tampilan Detail Komponen Perangkat keras. Implementasi Perangkat keras pada tiap *node* dilakukan dengan menghubungkan nRF24L01 dengan Arduino UNO, baik menggunakan Kabel Jumper atau PCB.

5.2.3 Implementasi Perangkat Lunak

Pada penelitian ini dibutuhkan adanya *library* pada sisi pemrograman perangkat lunak agar sistem dapat berjalan sesuai fungsinya. Fungsi *library* sendiri untuk menunjang kinerja atau fungsionalitas dari komponen-komponen yang sudah dirancang. Salah satu fungsi adanya *library* dalam penelitian ini adalah agar sistem mampu mendapatkan data inputan yang berupa *string* dan mampu mengirimkan data tersebut. *Library* akan dimasukan ke dalam program yang tertanam pada perangkat pengolah data, yaitu Arduino UNO. Implementasi *library code* pada sisi *Node base* dapat dilihat pada Tabel 5.6. Implementasi *library code* pada sisi *Node client* dapat dilihat pada Tabel 5.7.

Tabel 5.6 *Library Node Base*

Library node base	
1	`#include <SPI.h>`
2	`#include <Mirf.h>`
3	`#include <nRF24L01.h>`
4	`#include <MirfHardwareSpiDriver.h>`
5	`#include <stdio.h>`
6	`#include <EEPROM.h>`

Library <SPI.h> digunakan untuk mengaktifkan pin SPI pada modul komunikasi *wireless* nRF24L01. *Library* <Mirf.h> digunakan untuk melakukan pengiriman data

pada modul komunikasi *wireless* nRF24L01. *Library* <Stdio.h> digunakan untuk melakukan perintah *input-output* pemrograman pada Bahasa C atau C++. *Library* <nRF24L01.h> dan <MirfHardwareSpiDriver.h> digunakan untuk mensupport *Library* <Mirf.h>. *Library* <EEPROM.h> digunakan untuk menyimpan data agar tidak hilang.

Tabel 5.7 *Library Node Client*

Library Node Client	
1	`#include <SPI.h>`
2	`#include <Mirf.h>`
3	`#include <nRF24L01.h>`
4	`#include <MirfHardwareSpiDriver.h>`

Library <SPI.h> digunakan untuk mengaktifkan pin SPI pada modul komunikasi *wireless* nRF24L01. *Library* <Mirf.h> digunakan untuk melakukan pengiriman data pada modul komunikasi *wireless* nRF24L01. *Library* <Stdio.h> digunakan untuk melakukan perintah *input-output* pemrograman pada Bahasa C atau C++. *Library* <nRF24L01.h> dan <MirfHardwareSpiDriver.h> digunakan untuk mensupport *Library* <Mirf.h>.

5.2.3.1 Implementasi pada *Node Base*

Pada *node base* ada *main program* dimana di *main program* tersebut terdapat beberapa fungsi lagi yang akan dijalankan secara berurutan. Implementasi *main program* pada sisi *node base* dapat dilihat pada Tabel 5.8.

Tabel 5.8 Pendefinisian Variabel *Node Base*

Kode Sumber Variabel *Node Base*	
1	`#define ID  1`
2	`#define nodeClient 4`
3	`#define ulangRouting  1`
4	`#define ulangAodvSession 1`
5	`String hasilRouting[nodeClient+1][2];`
6	`boolean active[nodeClient+1];`
7	`double millisNow;`
8	`char data[32];`
9	`char dataKirim[32];`
10	`String valSplitNode[10];`
11	`char x[9];`
12	`char xx[2];`
13	`int n;`
14	`String valSplit[7];`
15	`int banyakNodeYgDiLalui;`
16	`boolean flag = false;`
17	`boolean startAodv = true;`

```
18  char readSerial;
19  boolean startRREQ = false;
20  boolean startMessage = false;
21  boolean startNode = false;
22  String dataSerial;
23  String sendDataString;
24  boolean afterRouting = false;
```

Proses pertama ada pendefinisian variabel global yang akan digunakan pada *Main Program* untuk mengimplementasikan sistem, bisa dilihat pada baris 1-24 di Tabel 5.8. Pendefinisian tersebut meliputi: pendefinisian ID *node* pengirim, total *node* yang digunakan, berapa kali percobaan perulangan *routing*, berapa kali percobaan perulangan sesi AODV, menampung data hasil *routing* dalam bentuk matrix, mencatat jumlah *node* yang aktif, menyimpan waktu sekarang, menyimpan dan mengirim data max. 32 char., pemisah data dalam variabel *string array*, menyimpan jumlah *node* yang dilalui, indikator memulai RREQ dan menunggu kiriman paket RREP, indikator memulai AODV, menyimpan data serial, dan memulai pengiriman pesan.

Tabel 5.9 Memulai Instruksi Program di *Node Base*

Kode Sumber Instruksi Program *Node Base*	
25	`void setup(){`
26	`  Serial.begin(9600);`
27	`  Mirf.spi = &MirfHardwareSpi;`
28	`  Mirf.init();`
29	`  Mirf.setRADDR((byte *)"2222");`
30	`  Mirf.payload = 32;`
31	`  Mirf.channel = 99;`
32	`  Mirf.config();`
33	`  Serial.println("Beginning ... ");`
34	`  Serial.println("*** MENU ***");`
35	`  Serial.println("Type Q= Start Sending RREQ");`
36	`  Serial.println("Type T= Show Table Routing");`
37	`  Serial.println("Type M= Input Message For Send With AODV (format: message+! example: hola!)");`
38	`}`

Lalu berikutnya kita akan masuk ke tampilan awal program di Serial Monitor pada baris 25-38 di Tabel 5.9. Pada bagian void setup diatas, akan mengeksekusi perintah satu kali jalan. Fungsi void setup() akan melakukan pengaturan dasar seperti mengaktifkan fungsi *Library* yang digunakan, mengatur *baudrate*, *pipe* dan *channel* yang digunakan, serta memulai instruksi. Adapun instruksi yang tampil adalah: Beginning (Persiapan Munculnya Set. Instruksi), Type Q = Start Sending RREQ (Ketik “Q” untuk memulai proses discovery dan Route

Request), Type T= Show Table Routing (Ketik “T” untuk menampilkan Table Routing Tujuan yang sudah terbentuk setelah proses RREQ dan RREP selesai), dan Type M= Input Message For Send With AODV (Ketik “M” untuk Mengirimkan Pesan setelah proses AODV selesai dengan format: pesan+! , untuk membedakan pesan tersebut dengan paket sebelumnya).

Tabel 5.10 Proses Routing di *Node Base*

Kode Sumber Proses Routing Node Base	
39	`void loop(){`
40	`  if(Serial.available())`
41	`  {`
42	`  readSerial = Serial.read();`
43	`  if(readSerial == 'Q')`
44	`  {`
45	`  startRREQ = true;`
46	`  }`
47	`  if(readSerial == 'T')`
48	`  {`
49	`  Serial.println("Please Wait... \nShow Table Routing");`
50	`  showTblRouting();`
51	`  Serial.println("*** End Table Routing ***");`
52	`  }`
53	`  if(readSerial == 'M')`
54	`  {`
55	`  Serial.println("Input Message:");`
56	`  startMessage = true;`
57	`  }`
58	`  }`
59	`  while(startMessage)`
60	`  {`
61	`  if(Serial.available())`
62	`  {`
63	`  char cc = Serial.read();`
64	`  if(cc == '!')`
65	`  {`
66	`  valMessage(dataSerial);`
67	`  dataSerial = "";`
68	`  startMessage = false;`
69	`  startAodv = false;`
70	`  }`
71	`  else`
72	`  {`

```
73          dataSerial += cc;
74        }
75      }
76    }
77    while(startRREQ)
78    {
79      routingNode("hello",15);
80      Serial.println("*** PROSES RREQ IS DONE ***");
81      startRREQ = false;
82      if(afterRouting)
83      {
84        startAodv = false;
85      }
86    }
87    while(!startAodv)
88    {
89      afterRouting = true;
90      int panjangDataString = sendDataString.length() + 1;
91      char dataToChar[panjangDataString];
92      sendDataString.toCharArray(dataToChar, panjangDataString);
93          aodvSession(dataToChar,15);
94      Serial.println("Please Wait... Evaluasi Rute Node");
95      for(int evaluasiNode = 2; evaluasiNode <= nodeClient;
    evaluasiNode++)
96      {
97        showTblRouting();
98        if(active[evaluasiNode] == false)
99        {
100         Serial.print("Node ");
101         Serial.print(evaluasiNode);
102         Serial.println(" not connected \nRouting Ulang... \n");
103         startAodv = true;
104         startRREQ = true;
105         startMessage = false;
106       }
107     }
108     for(int y = 2; y <= nodeClient; y++)
109     {
110       active[y] = false;
111     }
112     }
113
```

114	}

Setelah mengeksekusi fungsi void setup, selanjutkan akan masuk pada fungsi void loop(), yang ada di baris 39-114. Pada fungsi void loop, program akan dieksekusi secara berkali-kali sampai ada perintah untuk memberhentikannya. Fungsi void loop() ini pertama akan menunggu *input user* dari Serial Monitor. Jika *user* menginput "Q", maka *node base* akan mengirim broadcast paket ke semua *node client* tetangganya dengan header paket "Q", begitu pun setiap *node client* akan meneruskan paket tersebut. Apabila semua *node client* tetangga sudah menerimanya maka mereka bersiap untuk mengirim kembali paket balasan menuju *node base* dengan paket header "P", mengirim *Route Reply*. Tiap paket yg dikirim akan disertai pesan "Hello" di dalamnya. Tiap sesi berjalan selama 15 detik, dan itu akan terus menerus berjalan untuk terus memastikan bahwa setiap *node* aktif. Jika ada salah satu *node* yang terputus, maka proses akan dimulai lagi dari awal, hingga muncul pesan yang menyatakan bahwa proses RREQ dan RREP selesai dijalankan. Jika *user* menginput "T", maka akan ditampilkanlah *node* yang aktif beserta rute terdekat menujunya. Ketika *user* menginputkan dalam kondisi salah satu *node* tidak aktif, maka slot yang ada akan kosong tampilannya, sambil memberi permintaan *routing* ulang, hingga *node* tersebut kembali aktif. Jika *user* menginput "M", maka pesan akan dikirim menuju *node* tujuan melalui *routing table* yang sudah terbentuk. Paket yang dikirim maksimal hanya terbatas 32 char.

Tabel 5.11 *Sending Data* di *Node Base*

Kode Sumber Sending Data Node Base	
1	`void sendingData(char dest[], char text[])`
2	`{`
3	`  Mirf.setTADDR((byte *)"2222");`
4	`  strcpy(dataKirim, "Q");`
5	`  strcat(dataKirim, ",");`
6	`  sprintf(x,"%d",ID);`
7	`  strcat(dataKirim, x);`
8	`  strcat(dataKirim, ",");`
9	`  strcat(dataKirim, dest);`
10	`  strcat(dataKirim, ",");`
11	`  strcat(dataKirim, text);`
12	`  strcat(dataKirim, ",");`
13	`  sprintf(x,"%d",ID);`
14	`  strcat(dataKirim, x);`
15	`  strcat(dataKirim, ",");`
16	`  Serial.print("Sending to: ");`
17	`  Serial.println(dataKirim);`
18	`  Mirf.send((byte *)&dataKirim);`
19	`    while(Mirf.isSending()){`
20	`  }`

```
}
void sendingAodv(char dest[], char text[], String route)
{
  Mirf.setTADDR((byte *)"2222");
  strcpy(dataKirim, "A");
  strcat(dataKirim, ",");
  sprintf(x,"%d",ID);
  strcat(dataKirim, x);
  strcat(dataKirim, ",");
  strcat(dataKirim, dest);
  strcat(dataKirim, ",");
  strcat(dataKirim, text);
  strcat(dataKirim, ",");
  String newRoute = "";
  newRoute = route;
  n = newRoute.length() + 1;
  x[n];
  newRoute.toCharArray(x, n);
  strcat(dataKirim, x);
  strcat(dataKirim, ",");
  Serial.print("Sending AODV to: ");
  Serial.println(dataKirim);
  Mirf.send((byte *)&dataKirim);
  while(Mirf.isSending()){
  }
}
void aodvSession(char text[], double timeOut)
{
  millis();
  for(int ulang = 0; ulang <= ulangAodvSession; ulang++)
  {
    for(int node = 2; node <= nodeClient; node++)
    {
      flag = false;
      millisNow = millis();
      while((millis()-millisNow) <= timeOut*1000)
      {
        if(!flag)
        {
          hasilRouting[node][0];
          //Serial.println(hasilRouting[node][0]);
```

```
62              parseNodeBefore(hasilRouting[node][0],">");
63              //Serial.println(valSplitNode[banyakNodeYgDiLalui]);
64                      String newRoute = "";
65              newRoute = valSplitNode[banyakNodeYgDiLalui];
66              n = newRoute.length() + 1;
67              xx[n];
68              newRoute.toCharArray(xx, n);
69              sendingAodv(xx,text,hasilRouting[node][0]);
70              receiveData(true);
71                       flag = true;
72            }
73            receiveData(true);
74          }
75        }
76      }
77    }
```

Tabel 5.11 adalah kode program detail mengenai sistem pengiriman data, yang mana merupakan awal proses *discovery route* hingga memulai menjalankan protokol AODV dan mengirimkan pesan dari *User* menuju *node* tujuan. Dimulai dengan membuka *port pipe* terlebih dahulu, maka *node base* akan mengirimkan paket *discovery* sekaligus menjalankan RREQ ke semua *node client* tetangganya, dengan format paket yang dikirim [Q, Node Pengirim, Node Tujuan, Hello, Jalur yang Dilewati], dimana "Q" merupakan paket Header yang menandakan paket merupakan paket RREQ. Setelah semua proses RREQ dan RREP selesai, dan Tabel Routing sudah ditentukan, maka Proses AODV akan dimulai, dimana proses ini akan menyeleksi jalur terpendek untuk mengirimkan data berdasarkan banyak *node* yang dilaluinya. Setelah ditemukan, maka *node base* akan mengirimkan pesan yang diinput oleh *user*. Dan disisipkan bersama paket AODV yang dikirim menuju node tujuan dengan format paket yang dikirim [A, Node Pengirim, Node Tujuan, Input Pesan dari User, Jalur yang Dilewati], dimana "A" merupakan paket Header yang menandakan paket merupakan paket kiriman AODV dan pesan dari *user*.

Tabel 5.12 *Receive Data* di *Node Base*

Kode Sumber *Receive Data Node Base*	
1	`void receiveData(boolean routingOrAodv)`
2	`{`
3	`  if(!Mirf.isSending() && Mirf.dataReady()){`
4	`    Mirf.getData((byte *)&data);`
5	`    //Serial.print("Data RAW: ");`
6	`    //Serial.println(data);`
7	`     parseData(data,",");`
8	`    if(!routingOrAodv)`

```
9       {
10        parseNodeBefore(valSplit[4],">");
11        cekDest(ID);
12      }
13      else
14      {
15        cekDestAodv(ID);
16      }
17    }
18  }
19  void parseData(char text[], String key)
20  {
21    String dataString = String(text);
22    int countSplitSecond=0;
23    int lastIndexSecond=0;
24    for(int j = 0; j < dataString.length(); j++)
25    {
26      if(dataString.substring(j, j+1) == key)
27      {
28
29  valSplit[countSplitSecond]=dataString.substring(lastIndexSecond,j);
30        lastIndexSecond = j + 1;
31        //Serial.print(countSplitSecond);
32        //Serial.print(":");
33        //Serial.println(valSplit[countSplitSecond]);
34        countSplitSecond++;
35      }
36    }
37  }
38  void cekDest(int myId)
39  {
40    if(valSplit[2] == String(myId))
41    {
42      if(valSplit[0] == "P")
43      {
44        Serial.println(data);
45        Serial.println("Received RREP");
46        tableRouting();
47        Serial.println();
48      }
49      else
```

```
50      {
51        //Serial.println("Got RREQ");
52      }
53    }
54    }
55  void cekDestAodv(int myId)
56  {
57    if(valSplit[2] == String(myId))
58    {
59      if(valSplit[0] == "V")
60      {
61        Serial.println("Received AODV");
62        showFromNodeData();
63        Serial.println();
64      }
65      else
66      {
67        //Serial.println("Got RREQ");
68      }
69    }
  }
```

Proses pertama ketika *node base* menerima paket dari *node client* adalah mengecek apakah *node base* sedang tidak dalam proses mengirim, jika tidak maka simpan dulu data yang diterima ke dalam *variable byte array* "data". Lalu kemudian panggil fungsi untuk *parsing* data yang diterima. Ini dilakukan agar *Node Base* bisa membedakan apakah data yg diterima merupakan proses RREP atau AODV. Cara membedakannya dengan mengecek header paket dan tujuan dari paket yang diterima.

Karena setelah semua *node client* mendapatkan paket RREQ dari *node base*, maka *node base* akan menerima paket kiriman balasan dari *node client* yang berformat [P, Node Pengirim, Node Tujuan, Hello, Jalur yang Dilewati], dimana “P” merupakan paket Header yang menandakan paket merupakan paket RREP, sembari menampilkan pesan *Received RREP.* Begitupun apabila paket konfirmasi yang diterima oleh *node base* adalah paket AODV beserta inputan pesan dari *user*, maka format paket yang diterima adalah [V, Node Pengirim, Node Tujuan, Input Pesan dari User, Jalur yang Dilewati], dimana “V” merupakan paket Header yang menandakan paket merupakan paket konfirmasi kiriman AODV dan pesan dari *user* yang telah sampai di *node* tujuan, sembari menampilkan di Serial Monitor pengirim, route, dan pesan input dari *user*.

Tabel 5.13 *Routing Node* di *Node Base*

Kode Sumber *Routing Node* di *Node Base*	
1	`void routingNode(char text[], double timeOut)`
2	`{`
3	`  millis();`
4	`  for(int ulang2x = 0; ulang2x<=ulangRouting; ulang2x++)`
5	`  {`
6	`    for(int node = 2; node <= nodeClient; node++)`
7	`    {`
8	`      flag = false;`
9	`      millisNow = millis();`
10	`      while((millis()-millisNow) <= timeOut*1000)`
11	`      {`
12	`        if(!flag)`
13	`        {`
14	`          String newRoute = "";`
15	`          newRoute = String(node);`
16	`          n = newRoute.length() + 1;`
17	`          xx[n];`
18	`          newRoute.toCharArray(xx, n);`
19	`                    sendingData(xx,text);`
20	`                    receiveData(false);`
21	`                    flag = true;`
22	`        }`
23	`        receiveData(false);`
24	`      }`
25	`    }`
26	`  }`
27	`}`

Tabel 5.11 diatas merupakan detail Fungsi *Routing node*, dimana proses *routing default* minimal mengirimkan dua kali pengiriman RREQ, dan akan terus dilakukan di semua *node* yang tersedia. Fungsi ini juga memberikan jeda *timeout* selama 15 detik. Hasil *routing* yang diperoleh disimpan di variabel char *array*. Fungsi ini juga memberi penanda untuk mulai mengirim dan menerima data.

Tabel 5.14 *Parsing ID* di *Node Base*

Kode Sumber *Parsing ID* di *Node Base*	
1	`void parseNodeBefore(String text, String key)`
2	`{`
3	`  String nodeId = String(text);`

```
4   nodeId += key;
5   int countSplitNode=0;
6   int lastIndexNode=0;
7   for(int j = 0; j < nodeId.length(); j++)
8   {
9     if(nodeId.substring(j, j+1) == key)
10    {
11      valSplitNode[countSplitNode] =
    nodeId.substring(lastIndexNode,j);
12      lastIndexNode = j + 1;
13      //Serial.println();
14      //Serial.print(countSplitNode);
15      //Serial.print(":");
16      //Serial.println(valSplitNode[countSplitNode]);
17      banyakNodeYgDiLalui = countSplitNode;
18      countSplitNode++;
19    }
20  }
21  //Serial.println("Node pengirim: ");
22  //Serial.println(banyakNodeYgDiLalui);
23 }
```

Fungsi dari *Parsing ID* ini digunakan untuk melihat perbedaan dari ID Pengirim, ID Tujuan, dan menghitung banyaknya node yang dilalui dengan cara pisah data yang diterima ke dalam sebuah variable *String Array*, yang akan melihat ada berapa *node* yang dilewati selain ID Milik Sendiri.

Tabel 5.15 Menampilkan *Table Routing* di Node *Base*

Kode Sumber *Table Routing Node Base*

```
1   void tableRouting()
2   {
3   int idNode = valSplitNode[banyakNodeYgDiLalui].toInt();
4     int panjangString = valSplit[4].length();
5     int panjangRouteBefore = hasilRouting[idNode][0].length();
6     if(panjangString < panjangRouteBefore || panjangRouteBefore ==
    0)
7     {
8       hasilRouting[idNode][0] = valSplit[4];
9       //Serial.println(hasilRouting[idNode][0]);
10    }
11    else
12    {
13
```

```
14      //Serial.println("Lebih Jauh");
15    }
16  }
17  void showTblRouting()
18  {
19    Serial.println("########## Table Routing #######");
20    for(int node = 2; node <= nodeClient; node++)
21    {
22      Serial.print("Node Ke ");
23      Serial.print(node);
24      Serial.print(" ");
25      Serial.println(hasilRouting[node][0]);
26    }
27    Serial.println("########## ------------ #######");
28  }
29  void showFromNodeData()
30  {
31    Serial.println("########## Data AODV #######");
32    Serial.print("Data RAW: ");
33    Serial.println(data);
34    Serial.print("From    :");
35    Serial.println(valSplitNode[banyakNodeYgDiLalui]);
36    Serial.print("Route   :");
37    Serial.println(valSplit[4]);
38    Serial.print("Message :");
39    Serial.println(valSplit[3]);
40    Serial.println("########### END ################");
41    int nodeActive = valSplitNode[banyakNodeYgDiLalui].toInt();
42    active[nodeActive] = true;
      }
```

void TableRouting() ini adalah fungsi untuk menampilkan *Routing Table* yang telah terbentuk setelah semua proses telah selesai dijalankan. Fungsi ini bekerja dengan membandingkan hasil *routing* yang terbaru dengan hasil *routing* sebelumnya, dan mencari jalur terpendek diantara hasil *routing* yang disimpan. *Table Routing* ini akan terus menampilkan info *routing table* dari *node* yang aktif. Fungsi ini juga bertanggungjawab menampilkan hasil akhir program beserta pesan yang diinputkan oleh user.

Tabel 5.16 *ValMessage* di Node Base

Kode Sumber *ValMessage Node Base*	
1	`void valMessage(String dataReceived)`

```
2   {
3     dataReceived += "#";
4     int countSplitSecond=0;
5     int lastIndexSecond=0;
6     for(int j = 0; j < dataReceived.length(); j++)
7     {
8       if(dataReceived.substring(j, j+1) == "#")
9       {
10        sendDataString                                          =
    dataReceived.substring(lastIndexSecond,j);
11
12        lastIndexSecond = j + 1;
13        Serial.println(sendDataString);
14        countSplitSecond++;
15      }
16    }
  }
```

Fungsi ini digunakan untuk menyimpan *input* pesan dari *user* menuju *node* tujuan, dan memberi akhiran "#" sebagai penanda akhir pesan. Dan menyimpan pesan *input* tersebut di *variable* sendDataString. Selain itu, fungsi ini juga memiliki fitur Parsing Message, yang membuat fungsi ini saling berkaitan dengan fungsi Parsing ID.

5.2.3.2 Implementasi pada *Node Client*

Sama seperti *node base*, pada *node client* pun ada *main program* dimana di *main program* tersebut terdapat beberapa fungsi lagi yang akan dijalankan secara berurutan. Implementasi *main program* pada sisi *node client* dapat dilihat pada Tabel 5.15.

Tabel 5.17 *Pendefinisian Variabel Node Client*

Kode Sumber Variabel *Node Client*	
1	`#define ID  2`
2	`char data[32];`
3	`String valSplit[7];`
4	`char dataKirim[32];`
5	`String valSplitNode[10];`
6	`char x[32];`
7	`int n;`
8	`int banyakNodeYgDiLalui;`
9	`String IdNextNodeRREP;`
10	`int longRute = 0;`
11	`int samePackage = 0;`

Proses pertama ada pendefinisian variabel global yang akan digunakan pada *Main Program* untuk mengimplementasikan sistem pada *node client*, bisa dilihat pada baris 1-11. Pendefinisian tersebut meliputi: pendefinisian ID *node* pengirim, menyimpan dan mengirim data max. 32 char., pemisah data dalam variabel *string array*, menyimpan jumlah *node* yang dilalui, melanjutkan kiriman *broadcast* jika bukan ID tujuan paket, panjang rute, dan melihat ID paket yang apabila sama dengan sebelumnya, maka akan dilewatkan saja tanpa *dibroadcast*.

Tabel 5.18 Memulai Program di *Node Client*

Kode Sumber Memulai Program *Node Client*	
12	`void setup(){`
13	`  Serial.begin(9600);`
14	`  Mirf.spi = &MirfHardwareSpi;`
15	`  Mirf.init();`
16	`  Mirf.setRADDR((byte *)"2222");`
17	`  Mirf.payload = 32;`
18	`  Mirf.channel = 99;`
19	`  Mirf.config();`
20	`  Serial.println("Listening...");`
21	`}`
22	`void loop(){`
23	`  receiveData();`
24	`}`

Lalu berikutnya pada baris 12-21. Sama seperti di *node base*, ada bagian void setup, yang akan mengeksekusi perintah satu kali jalan. Fungsi void setup() akan melakukan pengaturan dasar seperti mengaktifkan fungsi *Library* yang digunakan, mengatur *baudrate*, *pipe* dan *channel* yang digunakan, serta menampilkan pesan siap menerima kiriman paket data.

Setelah mengeksekusi fungsi void setup, selanjutkan akan masuk pada fungsi void loop(), yang ada di baris 22-24. Pada fungsi void loop, program memulai untuk menerima kiriman paket data, baik dari *node base* langsung atau dari *node client* tetangganya.

Tabel 5.19 *Receive* Data di *Node Client*

Kode Sumber Receive Data Node Client	
1	`boolean validData = true;`
2	`String dataSebelumnya;`
3	`boolean aodvSession = false;`
4	`void receiveData()`
5	`{`
6	`  if(!Mirf.isSending() && Mirf.dataReady()){`

```
7       validData = true;
8       samePackage = 0;
9       longRute = 0;
10          Mirf.getData((byte *)&data);
11      Serial.print("Data RAW: ");
12      Serial.println(data);
13      if(dataSebelumnya != data)
14      {
15        parseData(data,",");
16        parseNodeBefore(valSplit[4],">");
17        cekDest(ID);
18      }
19      else
20      {
21        Serial.println("Data Sama...");
22        for(int dd=0; dd<=32; dd++)
23        {
24          data[dd] = '\0';
25        }
26        delay(1000);
27      }
28      dataSebelumnya = data;
29    }
30  }
31  void parseData(char text[], String key)
32  {
33    String dataString = String(text);
34    int countSplitSecond=0;
35    int lastIndexSecond=0;
36    for(int j = 0; j < dataString.length(); j++)
37    {
38      if(dataString.substring(j, j+1) == key)
39      {
40        valSplit[countSplitSecond]                                    =
41  dataString.substring(lastIndexSecond,j);
42        lastIndexSecond = j + 1;
43        //Serial.print(countSplitSecond);
44        //Serial.print(":");
45        //Serial.println(valSplit[countSplitSecond]);
46        countSplitSecond++;
47      }
```

```
48    }
49  }
50  void cekDest(int myId)
51  {
52    for(int ii=0; ii<=4; ii++)
53    {
54      if(valSplit[ii] == "")
55      {
56        validData = false;
57      }
58    }
59    if(validData)
60    {
61      if(valSplit[2] == String(myId))
62      {
63        if(valSplit[0] == "P")
64        {
65          cekPaket(ID);
66          if(samePackage != 2)
67          {
68            //delay(5);
69            putNodeBefore(ID);
70            sendingDataP("P");
71            Serial.println("Received RREP, and Broadcast RREP");
72            Serial.println();
73          }
74          else
75          {
76            Serial.println("Paket Sama...");
77            Serial.println();
78          }
79        }
80        else if(valSplit[0] == "A")
81        {
82          aodvSession = true;
83          Serial.println("Got AODV, Sending AODV Data ");
84          putNodeBefore(ID);
85          sendingDataP("V");
86          //sendingDataP("V");
87          Serial.println();
88        }
```

```
89        else
90        {
91          //delay(5);
92          Serial.print("Got RREQ, Sending RREP ");
93          Serial.println(valSplitNode[banyakNodeYgDiLalui]);
94          sendingDataLastP("P");
95          Serial.println();
96        }
97      }
98      else
99      {
100       //delay(5);
101       if(valSplit[0] == "A")
102       {
103         aodvSession=true;
104         Serial.println("Broadcast AODV");
105         Mirf.send((byte *)&data);
106             while(Mirf.isSending()){
107         }
108       }
109       else if(valSplit[0] == "V")
110       {
111         Serial.println("AODV ke Coordinator");
112       }
113       else
114       {
115         if(valSplit[0] == "Q" && valSplit[2] == "1")
116         {
117           Serial.println("Error..");
118         }
119         else
120         {
121           if(!aodvSession)
122           {
123             Serial.println("Broadcast...");
124             sendingDataQ();
125           }
126         }
127       }
128     }
129   }
```

```
130  }
131  void cekPaket(int myId)
132  {
133    String id = String(myId);
134    for(int rutePaket = 0; rutePaket <= longRute; rutePaket++)
135    {
136      if(valSplitNode[rutePaket] == id)
137      {
138        samePackage++;
139      }
140    }
141  }
```

Proses pertama ketika *node client* menerima paket dari *node base* adalah mengecek paket data tersebut apakah sudah pernah di terima atau belum. Jika belum, maka simpan dulu data yang diterima ke dalam variable "dataSebelumnya", yang akan digunakan ketika menerima data lagi sebagai pembanding. Lalu Serial Monitor menampilkan paket data apa yang diterima, baik berupa data dengan paket header "Q" maupun "P".

Jika paket header "Q" yang diterima merupakan paket untuknya, maka kemudian *node* penerima akan langsung mengirimkan paket balasan dengan header "P", sembari serial monitor menampilkan "Got RREQ, Sending RREP (node tujuan)". Paket balasan tersebut diterima oleh *node* tetangganya dan mengecek apakah paket yang diterima ini adalah untuknya, jika tidak maka akan diteruskan hingga menuju kembali ke pengirim paket header "Q" dan serial monitor menampilkan "Received RREP, and Broadcast RREP". Setiap kali pengecekan selesai dan data yang diterima sama dengan sebelumnya, *payload* akan dikosongkan selama *delay* satu detik untuk persiapan menerima data baru. Begitupun apabila paket header yang diterima oleh *node* tujuan dari *node base* adalah "A" yang merupakan paket AODV beserta inputan pesan dari user, maka jika paket sudah diterima oleh *node* tujuan, *node* tujuan akan membalas dengan mengirim balik paket konfirmasi menuju *node base* dengan paket header "V". Serial monitor juga akan menampilkan "Got AODV, Sending AODV Data (node tujuan)".

Tabel 5.20 *Sending* Data di *Node Client*

Kode Sumber *Sending* Data *Node Client*	
1	`void sendingDataQ()`
2	`{`
3	`  String newRoute = "";`
4	`  Mirf.setTADDR((byte *)"2222");`
5	`  strcpy(dataKirim, "Q");`
6	`  strcat(dataKirim, ",");`

```
7     sprintf(x,"%d",ID);
8     strcat(dataKirim, x);
9     strcat(dataKirim, ",");
10    newRoute = valSplit[2];
11    n = newRoute.length() + 1;
12    x[n];
13    newRoute.toCharArray(x, n);
14    strcat(dataKirim, x);
15    strcat(dataKirim, ",");
16    newRoute = valSplit[3];
17    n = newRoute.length() + 1;
18    x[n];
19    newRoute.toCharArray(x, n);
20    strcat(dataKirim, x);
21    strcat(dataKirim, ",");
22    newRoute = valSplit[1] + ">" + String(ID);
23    n = newRoute.length() + 1;
24    x[n];
25    newRoute.toCharArray(x, n);
26    strcat(dataKirim, x);
27    strcat(dataKirim, ",");
28    Serial.println(dataKirim);
29    Serial.println("------------");
30      Mirf.send((byte *)&dataKirim);
31      while(Mirf.isSending()){
32    }
33  }
34  void sendingDataP(char text[])
35  {
36    String newRoute = "";
37    Mirf.setTADDR((byte *)"2222");
38    strcpy(dataKirim, text);
39    strcat(dataKirim, ",");
40    sprintf(x,"%d",ID);
41    strcat(dataKirim, x);
42    strcat(dataKirim, ",");
43    newRoute = IdNextNodeRREP;
44    n = newRoute.length() + 1;
45    x[n];
46    newRoute.toCharArray(x, n);
47    strcat(dataKirim, x);
```

```
48    strcat(dataKirim, ",");
49    newRoute = valSplit[3];
50    n = newRoute.length() + 1;
51    x[n];
52    newRoute.toCharArray(x, n);
53    strcat(dataKirim, x);
54    strcat(dataKirim, ",");
55    newRoute = valSplit[4];
56    n = newRoute.length() + 1;
57    x[n];
58    newRoute.toCharArray(x, n);
59    strcat(dataKirim, x);
60    strcat(dataKirim, ",");
61    Serial.println("teruskan : ");
62    Serial.println(dataKirim);
63    Mirf.send((byte *)&dataKirim);
64      while(Mirf.isSending()){
65    }
66  }
67  void sendingDataLastP(char text[])
68  {
69    String newRoute = "";
70    Mirf.setTADDR((byte *)"2222");
71    strcpy(dataKirim, text);
72    strcat(dataKirim, ",");
73    sprintf(x,"%d",ID);
74    strcat(dataKirim, x);
75    strcat(dataKirim, ",");
76     newRoute = valSplitNode[banyakNodeYgDiLalui];
77    n = newRoute.length() + 1;
78    x[n];
79    newRoute.toCharArray(x, n);
80    strcat(dataKirim, x);
81    strcat(dataKirim, ",");
82    newRoute = valSplit[3];
83    n = newRoute.length() + 1;
84    x[n];
85    newRoute.toCharArray(x, n);
86    strcat(dataKirim, x);
87    strcat(dataKirim, ",");
88    newRoute = valSplit[4] + ">" + String(ID);
```

```
89    n = newRoute.length() + 1;
90    x[n];
91    newRoute.toCharArray(x, n);
92    strcat(dataKirim, x);
93    strcat(dataKirim, ",");
94    Mirf.send((byte *)&dataKirim);
95      while(Mirf.isSending()){
96    }
97  }
```

Fungsi sendingData pada tabel 5.17 diatas untuk meneruskan paket data yang masuk yang bukan untuk *node* tersebut. Fungsi tersebut akan mengecek dan mencarikan rute untuk menuju *node* tujuan. Bisa berfungsi juga untuk mengirim *broadcast* pada node tetangga dengan paket header "Q" atau paket header "P" dengan cara membuka *port pipe* mereka lalu melihat dan menyimpan rute yang sudah tercantum pada paket yang dikirim lalu kemudian meneruskannya menuju *node* tujuan. Fungsi ini juga bertanggungjawab menerima dan meneruskan paket header "A" dan "V" yang sampai di *node* tersebut.

Tabel 5.21 *Parsing ID* di *Node Client*

Kode Sumber *Parsing ID* di *Node Client*

```
1   void parseNodeBefore(String text, String key)
2   {
3     String nodeId = String(text);
4     nodeId += key;
5     int countSplitNode=0;
6     int lastIndexNode=0;
7     for(int j = 0; j < nodeId.length(); j++)
8     {
9       if(nodeId.substring(j, j+1) == key)
10      {
11        valSplitNode[countSplitNode]                                    =
    nodeId.substring(lastIndexNode,j);
12        lastIndexNode = j + 1;
13        //Serial.println();
14        //Serial.print(countSplitNode);
15        //Serial.print(":");
16        //Serial.println(valSplitNode[countSplitNode]);
17        banyakNodeYgDiLalui = countSplitNode;
18        countSplitNode++;
19      }
20    }
21    longRute = countSplitNode;
22
```

```
23    //Serial.println();
24    //Serial.println(banyakNodeYgDiLalui);
25  }
26  void putNodeBefore(int myID)
27  {
28    String idString = String(myID);
29    for(int node = 0; node <= banyakNodeYgDiLalui; node++)
30    {
31      if(valSplitNode[node] == idString)
32      {
33        IdNextNodeRREP = valSplitNode[node-1];
34        Serial.print("Node selanjutnya: ");
35        Serial.println(IdNextNodeRREP);
36      }
37    }
    }
```

Sama seperti pada *node base*, Fungsi dari *Parsing ID* pada *node client* ini juga digunakan untuk melihat perbedaan dari ID Pengirim, ID Tujuan, dan menghitung banyaknya *node* yang dilalui dengan cara pisah data yang diterima ke dalam sebuah variable *String Array*, yang akan melihat ada berapa *node* yang dilewati selain ID Milik Sendiri.

Yang berbeda hanya adanya fungsi tambahan void putNodeBefore(int myID). Fungsi tersebut bekerja untuk untuk mengambil nilai ID pada rute, yakni nilai sebelumnya dari nilai ID sendiri. Misalnya jika rute yang dilalui sama dengan ID milik sendiri (Contoh: data rute= 1>2>3>4, ID sendiri = 2) maka ID yg akan di ambil adalah 1). Fungsi ini digunakan untuk mengirim balik paket balasan RREQ, yakni RREP, dari *node* tujuan menuju ke *node* base. Fungsi ini akan memastikan bahwa paket RREP yang dikirim akan menempuh rute yg sama dengan RREQ, hanya berkebalikan rutenya, misal jika kita gunakan contoh data rute diatas, maka rute tempuh RREP adalah 4>3>2>1.

BAB 6 PENGUJIAN

Bab ini memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan. Pengujian dilakukan untuk mengetahui bahwa semua kebutuhan fungsional dan non fungsional yang dirancang sebelumnya sudah terpenuhi.

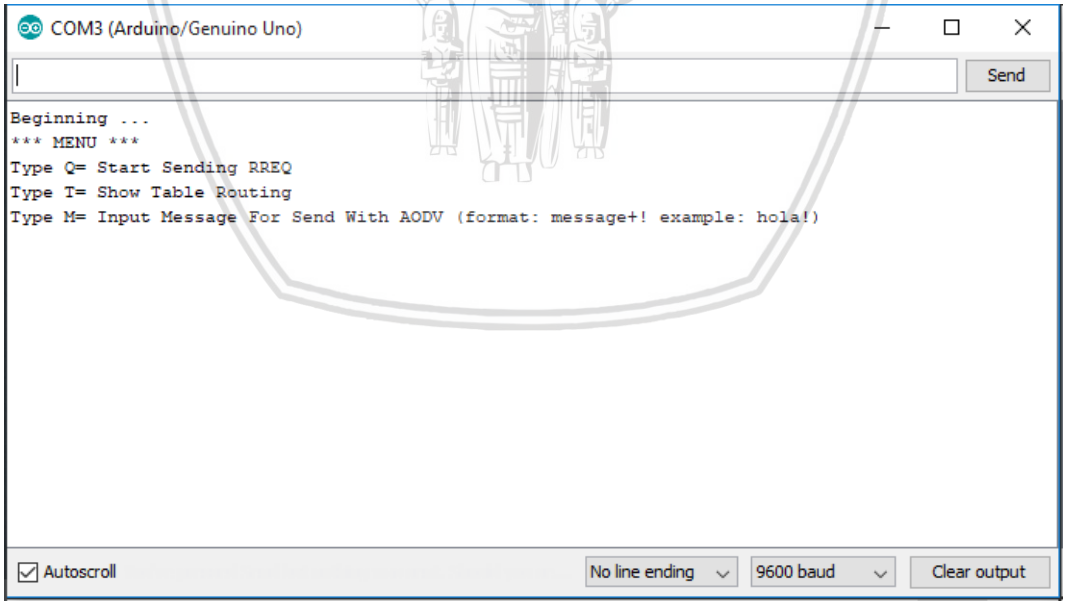
6.1 Pengujian Mengirim RREQ dan RREP

6.1.1 Tujuan

Pengujian ini bertujuan untuk mengetahui masing-masing node tetangga dan proses RREQ dan RREP telah berjalan dengan baik. Proses pengujian pada implementasi ini disebut *Discovery Phase* yang meliputi *broadcast*, RREQ, dan RREP.

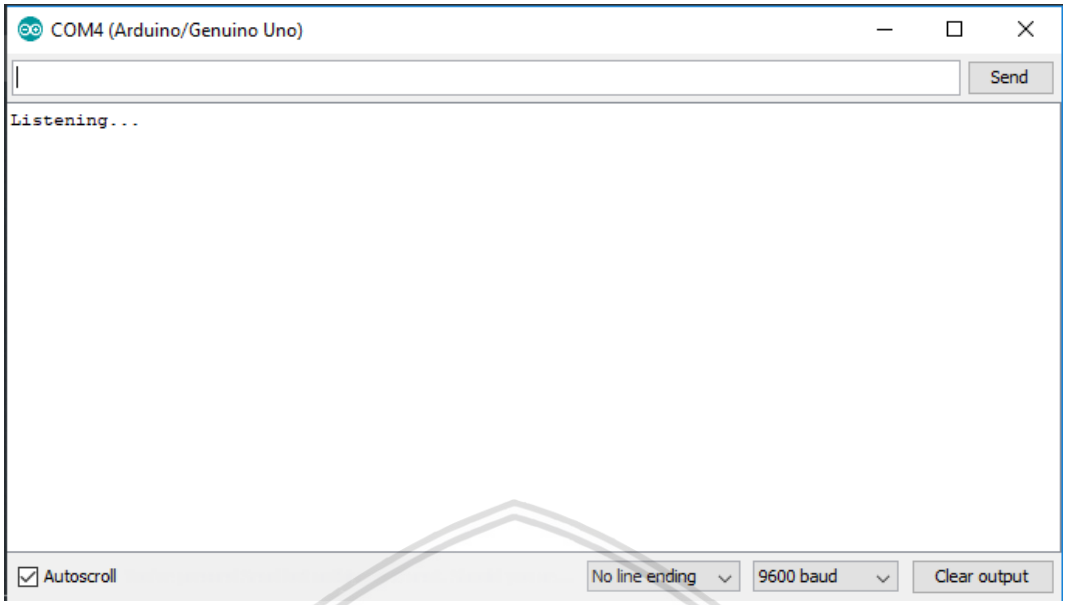
6.1.2 Prosedur Pengujian

Pengujian dilakukan dengan mengaktifkan 3 buah *node client* yang terhubung dengan 1 *node base* secara *wireless* menggunakan modul komunikasi *wireless* nRF24L01. Untuk prosedur pengujian, pertama pada sisi *node base* sebagai *Coordinator* diaktifkan terlebih dahulu dengan membuka *Serial Monitor* pada aplikasi Arduino. Kedua, pada masing-masing *node client* diaktifkan juga dengan membuka *serial monitor*. Setelah aktif, maka *user* akan mengetik pilihan yang tersedia di tampilan *serial monitor*, dalam hal ini hanya perlu mengetikkan "Q" pada Serial Monitor, maka proses akan berjalan.



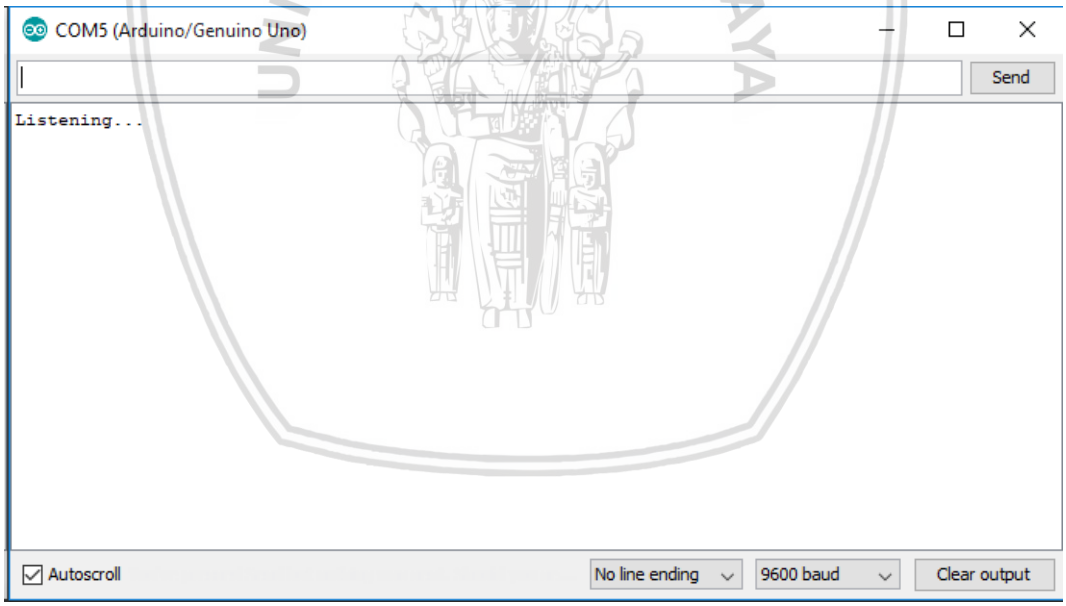
Gambar 6.1 Tampilan Awal Program Pada *Node Base*

Pada tampilan awal program di *Node base* seperti di Gambar 6.1, terdapat tampilan bahwa program telah dimulai dan menampilkan instruksi beserta keterangannya. Ketik "Q" untuk menjalankan *Discovery Phase* yang meliputi *broadcast*, RREQ, dan RREP.



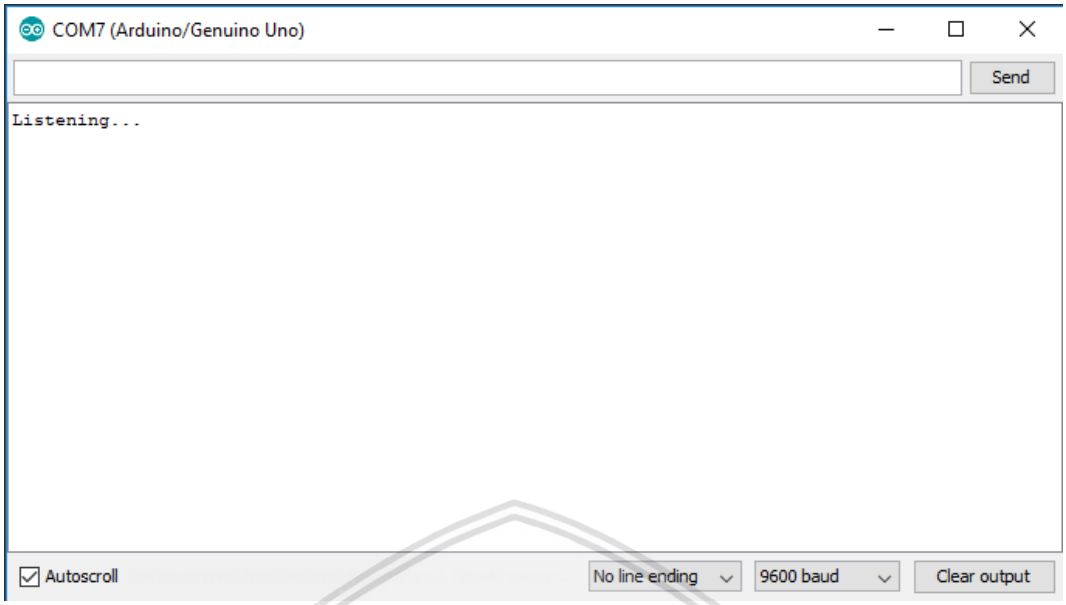
Gambar 6.2 Tampilan Awal Program Pada *Node Client* B

Pada tampilan awal program di *Node Client* B seperti di Gambar 6.2, *node client* sudah aktif dan bersiap untuk menerima kiriman paket data dari *node* tetangga.



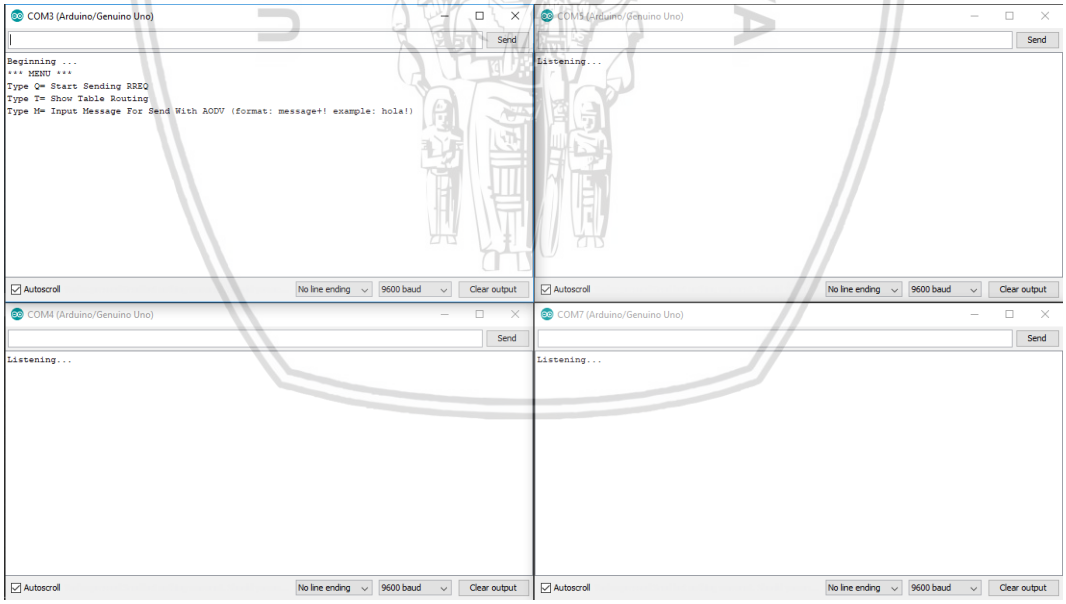
Gambar 6.3 Tampilan Awal Program Pada *Node Client* C

Pada tampilan awal program di *Node Client* C seperti di Gambar 6.3, *node client* sudah aktif dan bersiap untuk menerima kiriman paket data dari *node* tetangga.



Gambar 6.4 Tampilan Awal Program Pada *Node Client* D

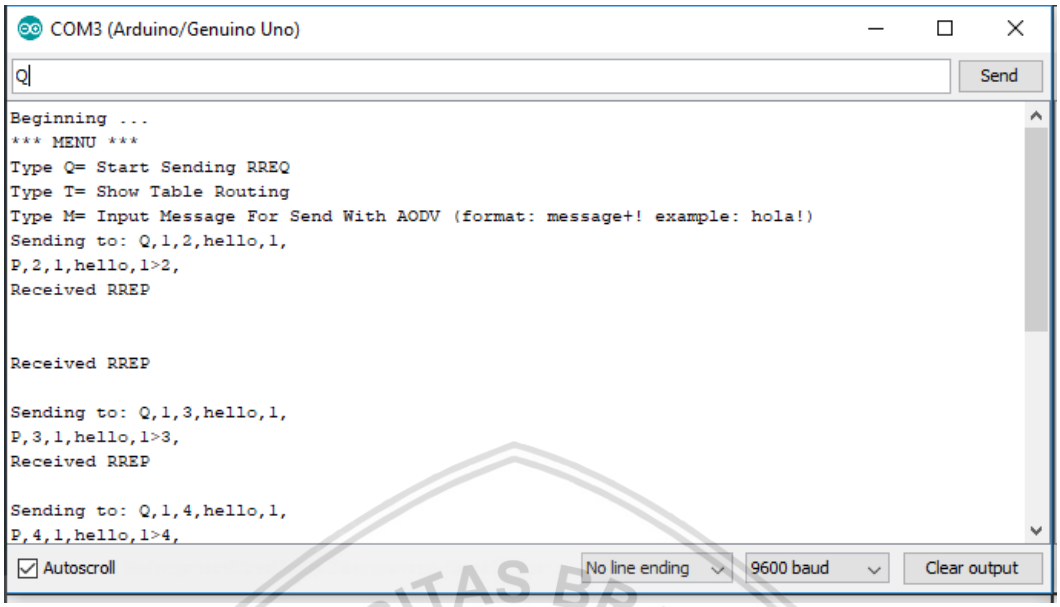
Pada tampilan awal program di *Node Client* D seperti di Gambar 6.4, *node client* sudah aktif dan bersiap untuk menerima kiriman paket data dari *node* tetangga.



Gambar 6.5 Tampilan Setiap Serial Monitor Disejajarkan untuk Diuji

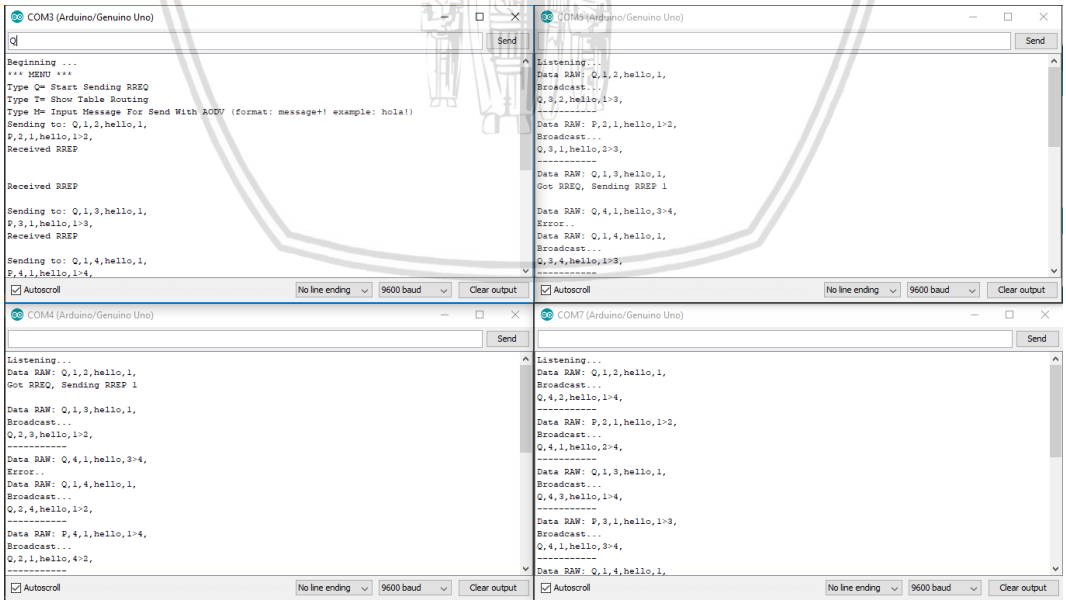
Semua tampilan serial monitor disejajarkan seperti tampilan pada Gambar 6.5 diatas, agar bisa terpantau semua paket data yang sedang diterima atau dilewatkan pada masing-masing *node*.

6.1.3 Hasil dan Analisis



Gambar 6.6 Proses *Discovery* Dimulai

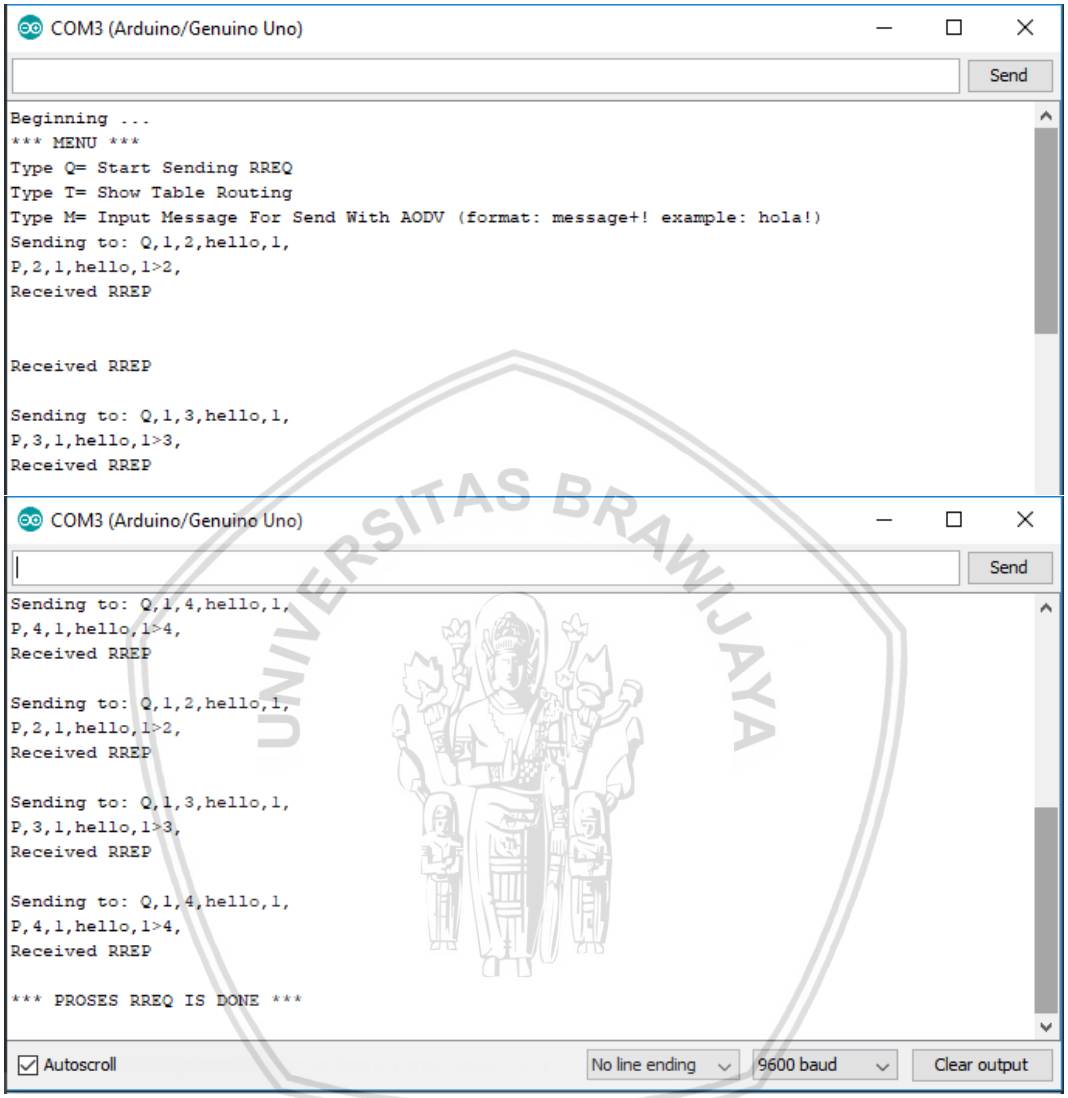
Proses *discovery* dimulai ketika *User* mengetikkan "Q" lalu tekan Enter pada Serial monitor pada *node base* seperti pada Gambar 6.6. Ini adalah perintah otomatis untuk mem*broadcast* paket untuk saling mengenali *node* tetangganya. Tidak hanya *node* base yang mengirim paket, antar *node* pun akan saling mengirim, menerima dan meneruskan paket yang didapatkannya.



Gambar 6.7 Tampilan Semua Ketika Proses *Discovery* Dimulai

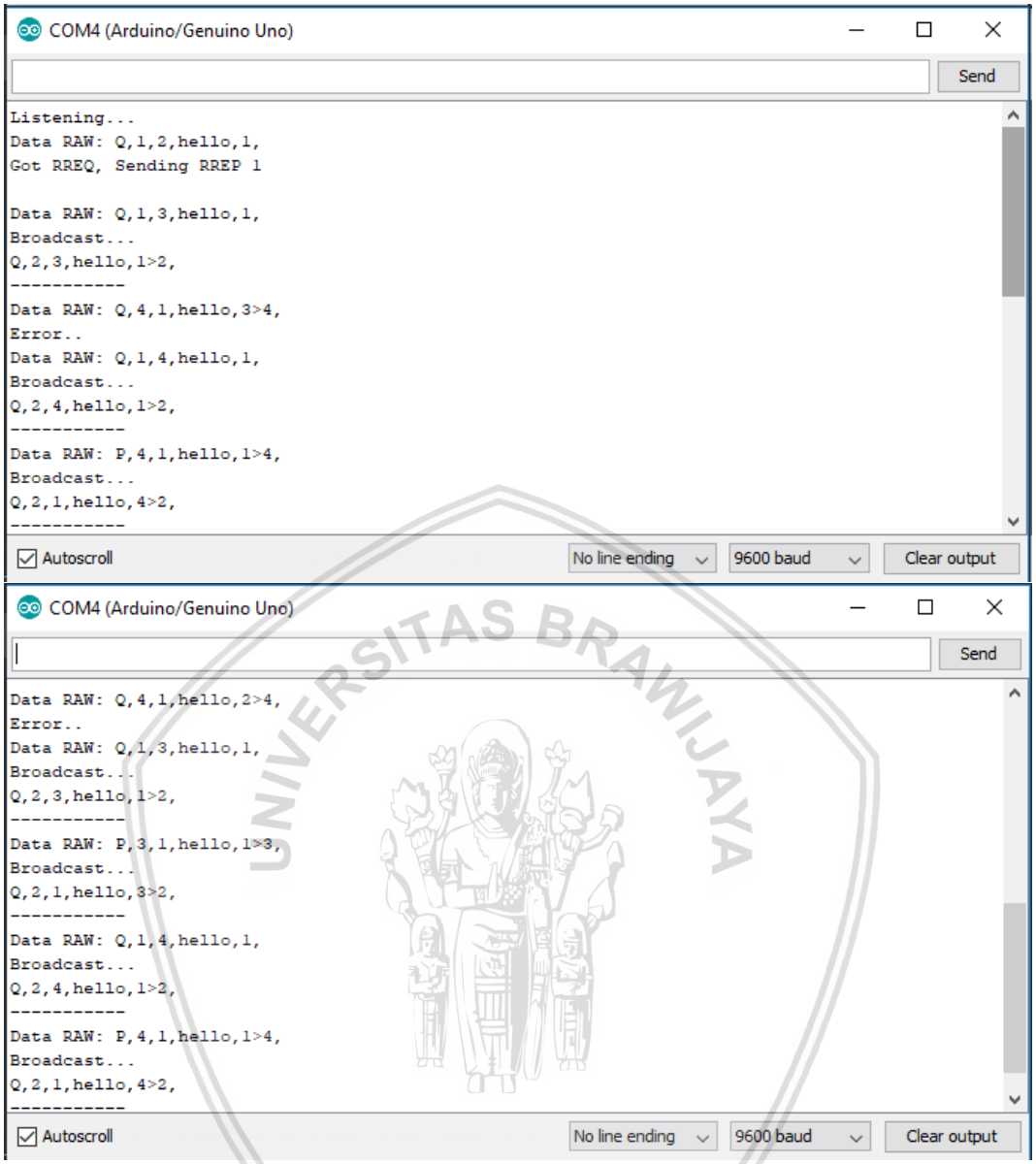
Ketika proses *discovery* sudah dimulai semua *node* akan saling mengirim paket dengan header "Q" untuk memberitahu *node* tetangganya. Jika paket yang datang itu memang untuk *node* tersebut, maka *node* tersebut akan mengirim balik

dengan paket header “P”. Jika bukan, maka paket akan diteruskan. Bisa dilihat pada Gambar 6.7.



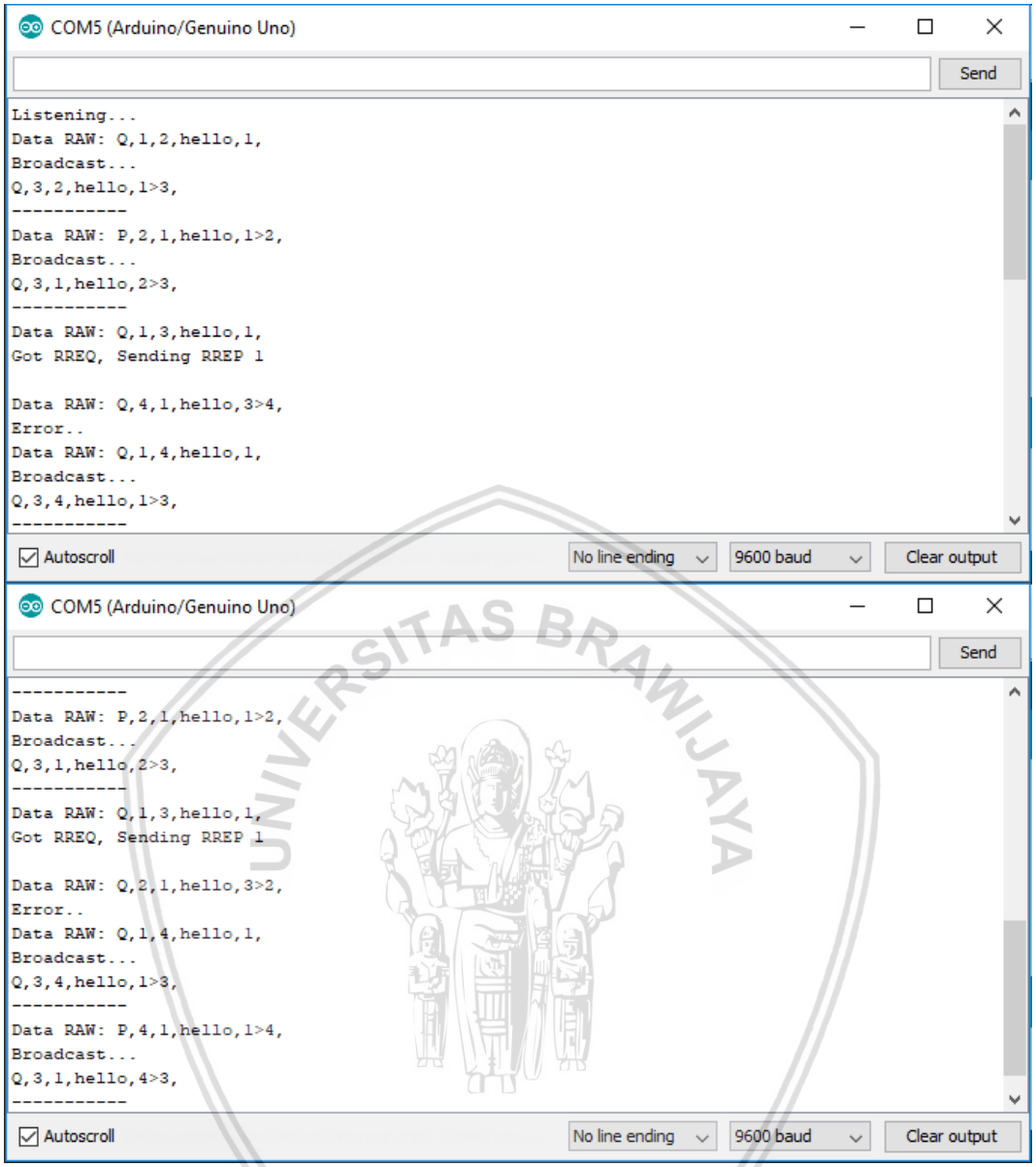
Gambar 6.8 Tampilan detail *node base* ketika Proses *Discovery*

Pada Gambar 6.8 diatas, *node base* mengirim *broadcast* paket dengan *header* “Q” ke semua *node* selain dirinya. *Node base* mengirim Paket RREQ ke *node client* B, *node client* B menerimanya dan langsung mengirim balik paket RREP ke *node base*. *Node base* mengirim Paket RREQ ke *node client* C, *node client* C menerimanya dan langsung mengirim balik paket RREP ke *node base*. *Node base* mengirim Paket RREQ ke *node client* D, *node client* D menerimanya dan langsung mengirim balik paket RREP ke *node base*. Jika paket yang diterima bukan untuk mereka, mereka akan meneruskannya ke *node* tetangganya.



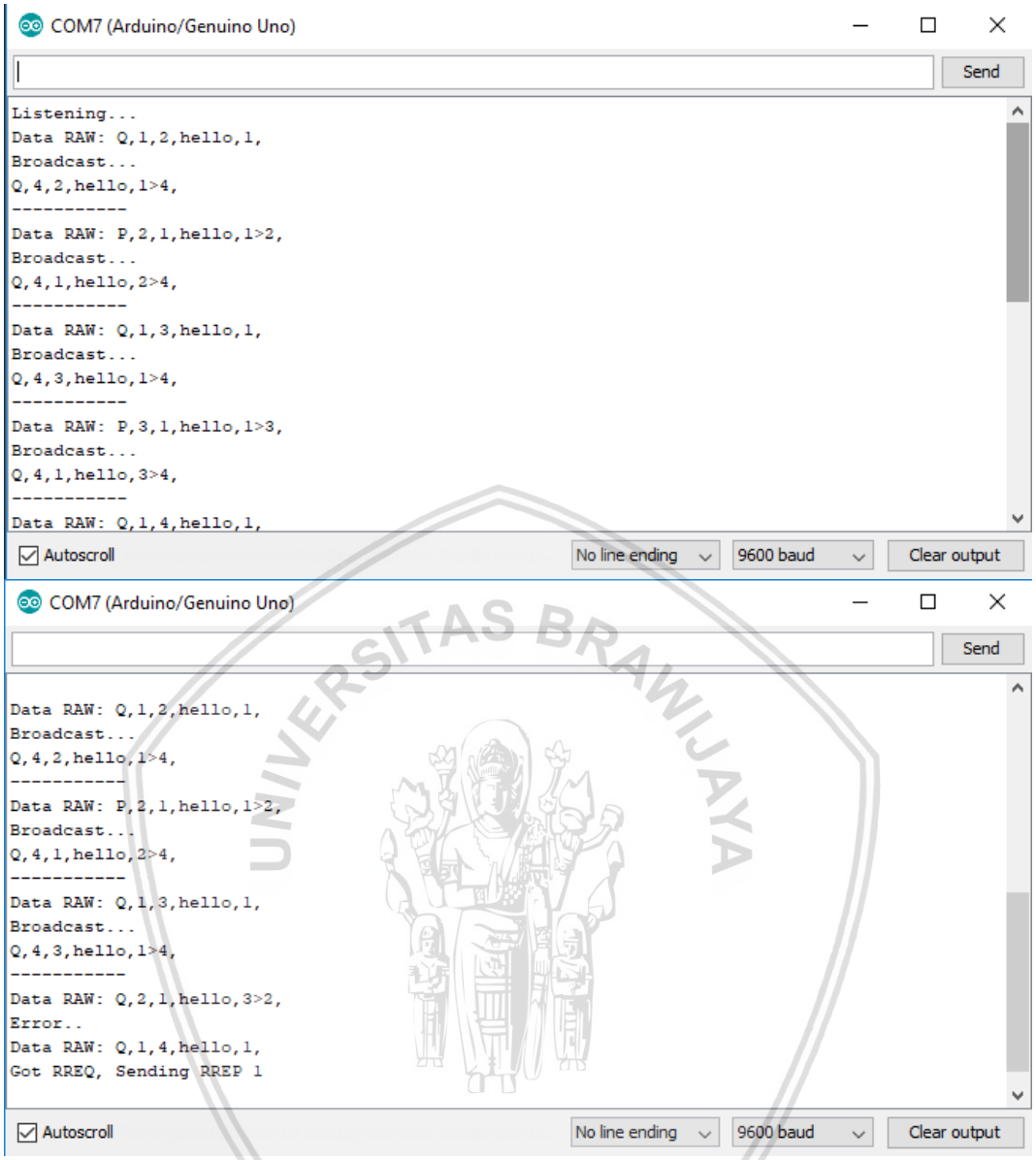
Gambar 6.9 Tampilan detail *node client* B ketika Proses *Discovery*

Pada Gambar 6.9 diatas, *node client* B menerima RREQ dari *node base* dan mengirim *broadcast* paket dengan *header* "Q" ke semua *node* lainnya. *Node base* mengirim Paket RREQ ke *node client* B, *node client* B menerimanya dan langsung mengirim balik paket RREP ke *node base*. *Node client* B mengirim Paket RREQ ke *node client* C, *node client* C menerimanya dan langsung mengirim balik paket RREP ke *node client* B. *Node client* B mengirim Paket RREQ ke *node client* D, *node client* D menerimanya dan langsung mengirim balik paket RREP ke *node client* B. Selain itu, *node client* B juga berperan menjadi jalur untuk melewatkan paket RREP dari *node client* C dan *node client* D menuju *node base*.



Gambar 6.10 Tampilan detail *node client* C ketika Proses *Discovery*

Pada Gambar 6.10 diatas, *node client* C menerima RREQ dari *node base* dan mengirim *broadcast* paket dengan *header* "Q" ke semua *node* lainnya. *Node base* mengirim Paket RREQ ke *node client* C, *node client* C menerimanya dan langsung mengirim balik paket RREP ke *node base*. *Node client* C mengirim Paket RREQ ke *node client* B, *node client* B menerimanya dan langsung mengirim balik paket RREP ke *node client* C. *Node client* C mengirim Paket RREQ ke *node client* D, *node client* D menerimanya dan langsung mengirim balik paket RREP ke *node client* C. Selain itu, *node client* C juga berperan menjadi jalur untuk melewatkan paket RREP dari *node client* B dan *node client* D menuju *node base*.



Gambar 6.11 Tampilan detail *node client* D ketika Proses *Discovery*

Pada Gambar 6.11 diatas, *node client* D menerima RREQ dari *node base* dan mengirim *broadcast* paket dengan *header* "Q" ke semua *node* lainnya. *Node base* mengirim Paket RREQ ke *node client* D, *node client* D menerimanya dan langsung mengirim balik paket RREP ke *node base*. *Node client* D mengirim Paket RREQ ke *node client* C, *node client* C menerimanya dan langsung mengirim balik paket RREP ke *node client* D. *Node client* D mengirim Paket RREQ ke *node client* B, *node client* B menerimanya dan langsung mengirim balik paket RREP ke *node client* D. Selain itu, *node client* D juga berperan menjadi jalur untuk melewatkan paket RREP dari *node client* B dan *node client* C menuju *node base*.

6.2 Pengujian Menampilkan *Table Routing*

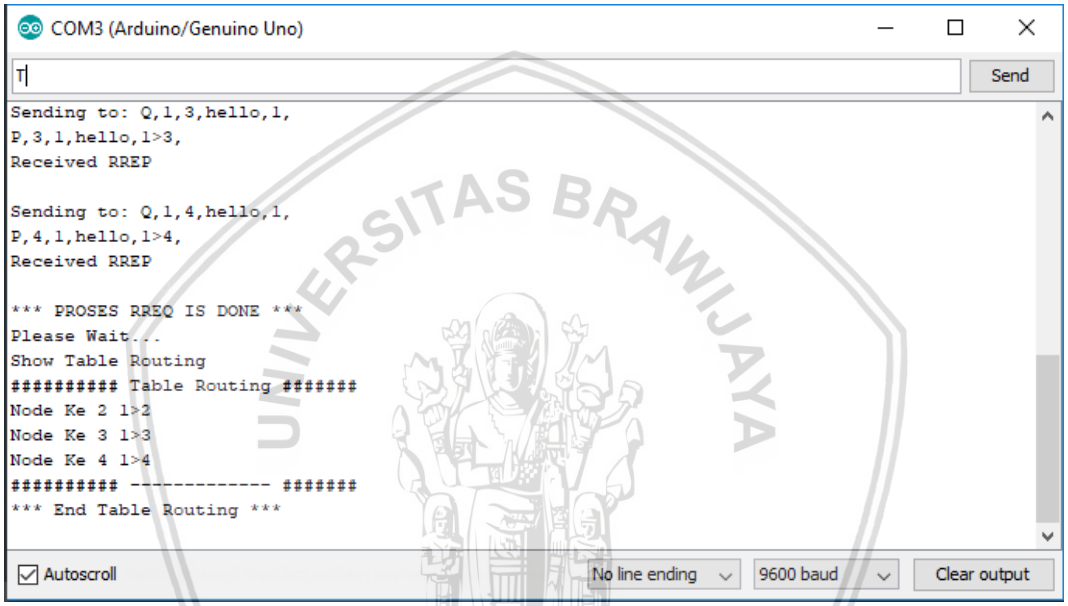
6.2.1 Tujuan

Pengujian ini bertujuan untuk menampilkan Table Routing yang terbentuk setelah proses *Discovery Phase* selesai.

6.2.2 Prosedur Pengujian

Setelah proses *Discovery Phase* selesai, maka *user* cukup mengetikkan "T" pada *serial monitor* untuk menampilkan *routing table* terpendek menuju tujuan.

6.2.3 Hasil dan Analisis



Gambar 6.12 Tampilan *Table Routing*

Setelah semua *node* sudah saling terhubung dan mengenali tetangganya, jika *user* ingin melihat jalur *routing* terpendek yang bisa ditempuh, maka *user* akan mengetikkan "T" untuk melihat tabel jalur *routing* yang bisa ditempuh untuk mengirim data. Contoh tampilan tabel jalur *routing* yang terbentuk bisa dilihat di Gambar 6.12

6.3 Pengujian Percobaan Mengirim Data

6.3.1 Tujuan

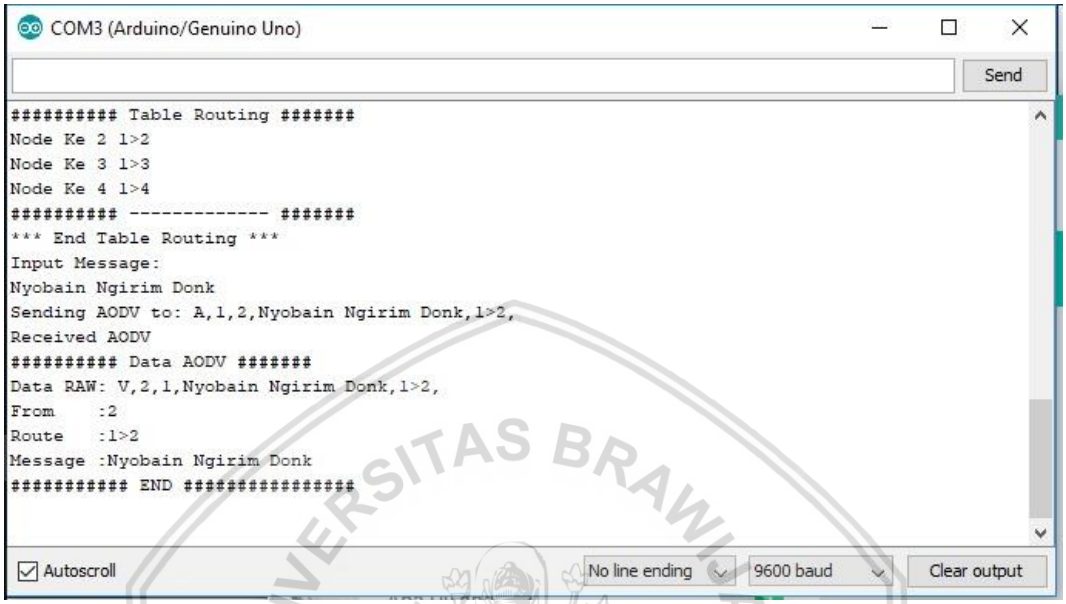
Pengujian ini bertujuan untuk mengirim data yang diinput oleh *user*. Proses pengujian pada implementasi ini disebut *Sending Data Phase* yang meliputi pengiriman data input dari *User* menggunakan route yang sudah diimplementasikan AODV.

6.3.2 Prosedur Pengujian

Pengujian ini dilakukan setelah *Discovery* Phase selesai dilakukan. Selanjutnya *user* ingin mengirim pesan menuju *node client* B misalnya, dalam

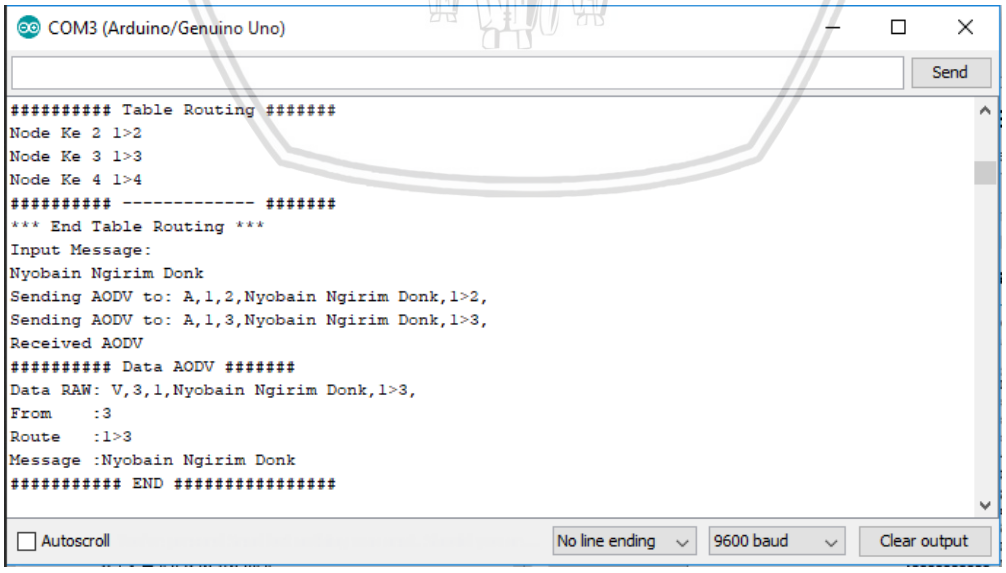
Gambar 6.13 dicontohkan *user* harus mengetikkan instruksi “M” lalu menekan Enter terlebih dahulu sebelum mengetikkan pesan “Nyobain Ngirim Donk!” pada serial monitor.

6.3.3 Hasil dan Analisis



Gambar 6.13 Percobaan mengirimkan pesan ke *node client* B

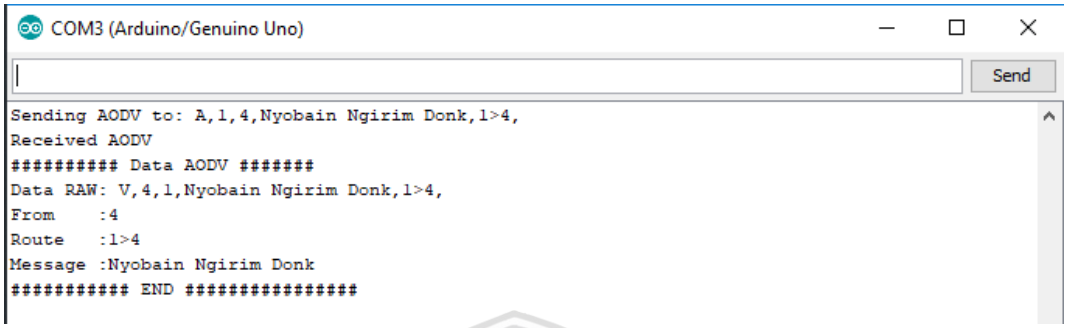
Pesan tersebut akan dikirim dengan protokol AODV yang ditandai dengan paket header “A” dari *node base* menuju *node client* B. *Node client* B akan menerimanya dan mengirim paket balasan dengan paket header “V” menuju *node base*, dan isi pesan kemudian akan ditampilkan pada serial monitor beserta pengirim dan *route* yang dilewati pada saat mengirimkan pesan tersebut.



Gambar 6.14 Percobaan mengirimkan pesan ke *node client* C

Pesan tersebut akan dikirim dengan protokol AODV yang ditandai dengan paket header “A” dari *node base* menuju *node client* C. *Node client* C akan

menerimanya dan mengirim paket balasan dengan paket header "V" menuju *node base*, dan isi pesan kemudian akan ditampilkan pada serial monitor beserta pengirim dan *route* yang dilewati pada saat mengirimkan pesan tersebut.



Gambar 6.15 Percobaan mengirimkan pesan ke *node client* D

Pesan tersebut akan dikirim dengan protokol AODV yang ditandai dengan paket header "A" dari *node base* menuju *node client* D. *Node client* D akan menerimanya dan mengirim paket balasan dengan paket header "V" menuju *node base*, dan isi pesan kemudian akan ditampilkan pada serial monitor beserta pengirim dan *route* yang dilewati pada saat mengirimkan pesan tersebut.

6.4 Pengujian Fungsionalitas Sistem

6.4.1 Tujuan

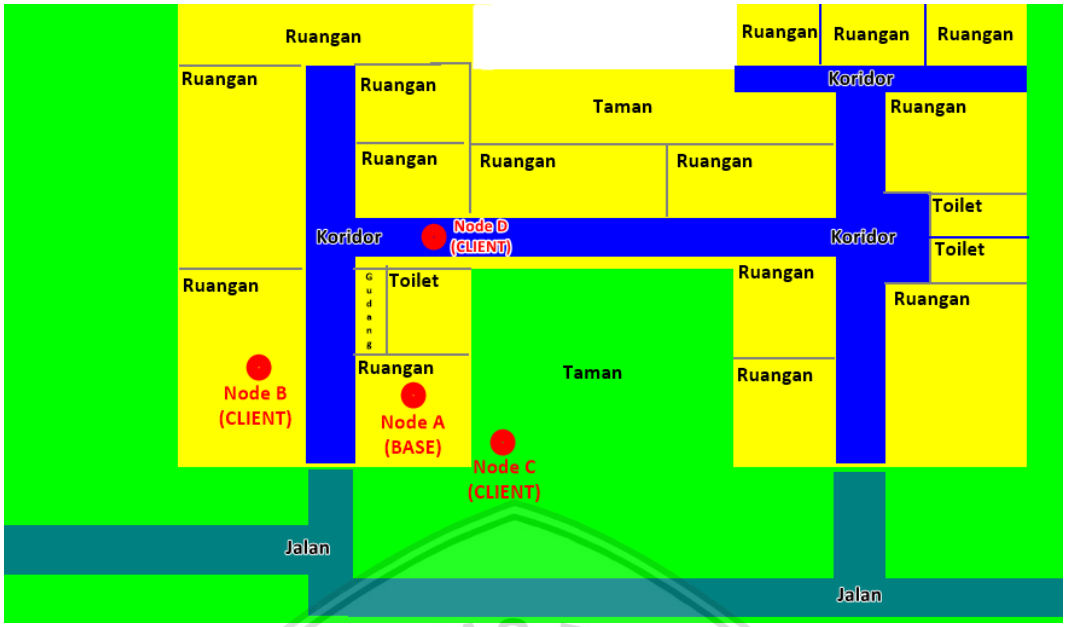
Pengujian fungsionalitas sistem dikatakan berhasil dilakukan apabila setiap node diam statis, node bergerak dinamis tanpa hambatan, dan node bergerak dinamis dengan hambatan sukses mengirimkan paket data menuju node tujuan.

6.4.2 Prosedur Pengujian

Pengujian dilakukan beruntun dengan urutan pengujian:

1. Pengujian Node Statis Tanpa Hambatan
2. Pengujian Node Dinamis Tanpa Hambatan
3. Pengujian Node Dinamis Dengan Hambatan

Masing-masing pengujian akan dilakukan berulang sebanyak 5 kali. Hasil dari pengujian tersebut akan ditampilkan dalam bentuk tabel yang menyatakan keberhasilan antar node saling terkoneksi, jalur routing terbentuk, dan pengiriman data input dari *user* dengan protokol AODV telah sukses terkirim. Khusus pada pengujian node dinamis dengan hambatan yang berupa pintu, dinding, dan kaca, maka denah lokasi penempatan tiap node bisa dilihat pada Gambar 6.16. Sementara pada pengujian node statis dan dinamis tanpa hambatan, lokasi tiap node berada pada satu ruangan yang sama, yakni berada pada ruangan node A (base) jika dilihat pada Gambar 6.16.



Gambar 6.16 Denah lokasi pengujian Node Dinamis Dengan Hambatan

6.4.3 Hasil dan Analisis

Tabel 6.1 Hasil Pengujian Node Statis Tanpa Hambatan

NO	Pengujian	Hasil
1	Pengujian ke-1	Berhasil Terkirim
2	Pengujian ke-2	Berhasil Terkirim
3	Pengujian ke-3	Berhasil Terkirim
4	Pengujian ke-4	Berhasil Terkirim
5	Pengujian ke-5	Berhasil Terkirim

Pada Tabel 6.1, saat pengujian node statis tanpa hambatan setiap node baik node *base* maupun node *client* dapat mengirim dan menerima paket data secara sempurna. Sehingga jika dihitung persentase keberhasilannya:

$$Keberhasilan = \frac{Data\ terkirim}{Total\ data} \text{x}\ 100\% = \frac{5}{5} \text{x}\ 100\% = 100\%$$

Tabel 6.2 Hasil Pengujian Node Dinamis Tanpa Hambatan

NO	Pengujian	Hasil
1	Pengujian ke-1	Berhasil Terkirim
2	Pengujian ke-2	Berhasil Terkirim
3	Pengujian ke-3	Berhasil Terkirim
4	Pengujian ke-4	Berhasil Terkirim
5	Pengujian ke-5	Berhasil Terkirim

Pada Tabel 6.2, saat pengujian node dinamis tanpa hambatan setiap node baik node *base* maupun node *client* dapat mengirim dan menerima paket data secara sempurna juga. Sehingga jika dihitung persentase keberhasilannya:

$$Keberhasilan = \frac{Data\ terkirim}{Total\ data} \text{x } 100\% = \frac{5}{5} \text{ x } 100\% = 100\%$$

Tabel 6.3 Hasil Pengujian Node Dinamis Dengan Hambatan

NO	Pengujian	Hasil
1	Pengujian ke-1	Berhasil Terkirim
2	Pengujian ke-2	Berhasil Terkirim
3	Pengujian ke-3	Gagal Terkirim
4	Pengujian ke-4	Gagal Terkirim
5	Pengujian ke-5	Berhasil Terkirim

Sementara itu pada Tabel 6.3, saat pengujian node dinamis dengan hambatan berupa pintu, dinding, dan kaca, node *base* maupun node *client* hanya dapat mengirim dan menerima paket data secara sempurna di pengujian pertama, kedua, dan kelima. Pada pengujian ketiga dan keempat, node client mengalami error sehingga tidak terdeteksi oleh node base. Sehingga jika dihitung persentase keberhasilannya:

$$Keberhasilan = \frac{Data\ terkirim}{Total\ data} \text{x } 100\% = \frac{3}{5} \text{ x } 100\% = 60\%.$$

Total semua skenario pengujian dilakukan sebanyak 15 kali, maka didapatkan persentase rata-rata keberhasilan pengujian sistem adalah:

$$Rata - rata\ keberhasilan = \frac{Pengujian\ berhasil}{Total\ pengujian} \text{x } 100\% = \frac{13}{15} \text{ x } 100\%$$
$$= \mathbf{86,6}\%.$$

BAB 7 KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan dan saran yang diperoleh dari rumusan masalah penelitian dengan melakukan analisis dan pengujian.

7.1 Kesimpulan

Berdasarkan hasil dari tahapan perancangan, implementasi, pengujian serta analisis hasil pengujian yang telah dilakukan, penulis dapat menarik kesimpulan bahwa:

1. Sistem *mobile ad-hoc network* dapat diimplementasikan pada sistem dengan merakit komponen-komponen yang telah ditentukan seperti Arduino UNO sebagai mikrokontroler dan nRF24L01. *Node base* dapat berkomunikasi dengan ketiga *node client* dengan menggunakan komunikasi *wireless* nRF24L01.

2. Protokol AODV sebagai metode *routing* protokol antar *node* berhasil diimplementasikan. Ketiga *node client* dan *node base* juga telah berhasil mengirim pesan dari proses pengujian yang dilakukan.

3. Dari pengujian sistem, semua *node* telah berhasil saling tehubung dan berkomunikasi serta dapat mengirimkan pesan yang diketikkan oleh *user*. Pada ketiga *node client* dan *node base* berhasil melakukan pengiriman dan penerimaan data sesuai dengan skenario yang telah dirancang. Hal ini membuktikan bahwa penerapan *mobile ad-hoc network* (manet) dengan menggunakan protokol AODV telah berhasil diimplementasikan pada perangkat berbasis nRF24L01.

4. Dari 15 kali pengujian fungsionalitas sistem, dimana skenarionya adalah node statis tanpa hambatan, node dinamis tanpa hambatan dan node dinamis dengan hambatan berupa pintu, dinding, atau kaca, semua *node* melakukan pengiriman dan penerimaan data dengan persentase keberhasilan sebesar 86,6%. Hal ini membuktikan bahwa pengujian fungsionalitas sistem telah berhasil diterapkan baik ada atau tidaknya hambatan.

7.2 Saran

Berdasarkan kesimpulan yang telah didapatkan, terdapat beberapa saran untuk penelitian selanjutnya sebagai langkah pengembangan sistem ini, yaitu:

1. Disarankan untuk menambahkan implementasi metode *routing* protokol yang lain agar dapat digunakan sebagai perbandingan.

2. Disarankan untuk menambahkan implementasi metode *routing* protokol yang lain agar pengiriman data lebih efisien.

3. Disarankan untuk melakukan penambahan *node client* agar menghasilkan implementasi metode *routing* protokol AODV yang lebih baik. Sehingga didapatkan hasil perbandingan *performance* pengiriman data oleh 3 *node client* dengan *"n"* node *client*.

4. Disarankan untuk mengimplementasikan dengan penambahan *input* dari sensor agar dapat lebih terlihat lagi fungsi dan kegunaannya.

5. Sistem ini dapat digunakan sebagai dasar untuk pembuatan sistem informasi pada daerah yang minim energi dan membutuhkan transfer data yang cepat.

DAFTAR PUSTAKA

Amilia, F., Marzuki & Agustina, 2014. Analisis Perbandingan Kinerja Protokol Dynamic Source Routing (DSR) dan Geographic Routing Protocol (GRP) pada Mobile Ad-Hoc Network (MANET). *Jurnal Sains, Teknologi dan Indonesia,* Vol. 12(No. 1), pp. 9-15.

Amitava, M., 2003. *Location Management and Routing in Mobile Wireless Networks,* Boston & London: Artech House.

Anon., n.d. [Online] Available at: https://store.arduino.cc/usa/arduino-uno-rev3 [Accessed 11 September 2017].

Anon., n.d. [Online] Available at: http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01 [Accessed 11 September 2017].

Benardi, B., 2009. *Analisa Unjuk Kerja Jaringan Nirkabel Ad-Hoc dalam Beberapa Situasi yang Berbeda Ditinjau dari Sudut Pandang Routing,* s.l.: Universitas Mercubuana.

Kopp, C., 2002. Background Article. In: *Ad-Hoc Networking.* s.l.:Published in 'System', pp. 33-40.

Larsson, T. & Hedman, N., 1998. *Routing Protocol in Wireless Ad-Hoc Networks a Simulation Study,* Stockholm: Lulea University of Technology.

Murthy, C., Siva, R. & Manoj, B., 2004. *Ad-Hoc Wireless Networks Architectures and Protocols,* USA: Pearson Education, Inc.

Nordic Semiconductor ASA, n.d. *Single Chip 2.4 GHz Transceiver.* [Online] Available at: http://data.mecheng.adelaide.edu.au/robotics/WWW_Devs/Dragon12/rtmc9S12Target/nRF24L01_prelim_prod_spec_1_2.pdf [Accessed 26 September 2017].

Perkins, C., Belding-Royer, E. & Das, S., 2003. Ad-Hoc on Demand Distance Vector (AODV) Routing. *IETF Network Working Group.*

Ramadhan, Y., Abdi, M. & Mike, R., 2011. *Analisa Performa Routing Protokol AODV, OLSR, dan DSDV menggunakan NS-3 pada Mobile Ad-Hoc Network,* Universitas Bina Nusantara: Skripsi.

Wehrle, K. & Gross, J., 2010. *Modelling and Tools for Network Simulation,* Germany: Springer.

Wismanu, 2004. *Evaluasi Unjuk Kerja Protokol Rute pada Jaringan Wireless Ad-Hoc Multi Hop,* s.l.: JTE-FTI.